Training Your Sparse Neural Network Better with Any Mask

Ajay Jaiswal [1] **Haoyu Ma** [2] **Tianlong Chen** [1] **Ying Ding** [1] **Zhangyang Wang** [1]

Abstract

Pruning large neural networks to create high-quality, independently trainable sparse masks, which can maintain similar performance to their dense counterparts, is very desirable due to the reduced space and time complexity. As research effort is focused on increasingly sophisticated pruning methods that leads to sparse subnetworks trainable from the scratch, we argue for an orthogonal, under-explored theme: *improving training techniques for pruned sub-networks, i.e. sparse training*. Apart from the popular belief that only the quality of sparse masks matters for sparse training, in this paper we demonstrate an alternative opportunity: one can *carefully customize the sparse training techniques to deviate from the default dense network training protocols*, consisting of introducing "ghost" neurons and skip connections at the early stage of training, and strategically modifying the initialization as well as labels. Our new sparse training recipe is generally applicable to improving training from scratch with various sparse masks. By adopting our newly curated techniques, we demonstrate significant performance gains across various popular datasets (CIFAR-10, CIFAR-100, TinyImageNet), architectures (ResNet-18/32/104, Vgg16, MobileNet), and sparse mask options (lottery ticket, SNIP/GRASP, SynFlow, or even randomly pruning), compared to the default training protocols, especially at high sparsity levels. Code is at https://github.com/VITA-Group/ToST.

1. Introduction

Deep neural networks (NN) have achieved significant progress in many tasks such as classification, detection, and segmentation. However, most existing models are computationally extensive and overparameterized, thus it is difficult to deploy these models in real-world devices. To address this issue, many efforts in direction of knowledge distillation (Hinton et al., 2015b), quantization (Hubara et al., 2018), and pruning (LeCun et al., 1989), have been devoted to compressing the heavy model into a lightweight counterpart. Among them, network pruning (LeCun et al., 1990; Han et al., 2015a;b; Li et al., 2016; Liu et al., 2019; Frankle & Carbin, 2018), which identifies sparse sub-networks (aka. sparse mask) by removing unnecessary connections, stands as one of the most effective methods.

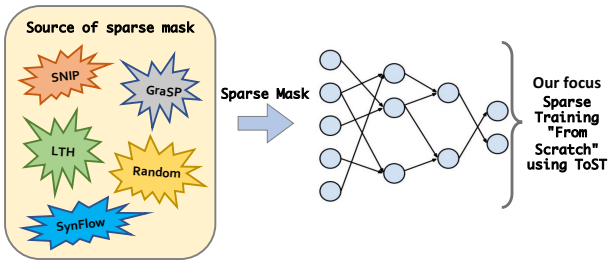

Figure 1. We aim to improve training of ANY sparse mask from scratch using our proposed sparse training toolkit (ToST).

Recently, a significant amount of research efforts have been focused towards developing increasingly sophisticated and efficient pruning algorithms (Lee et al., 2018; Wang et al., 2020; Frankle & Carbin, 2018; Frankle et al., 2019; Tanaka et al., 2020), to identify the sparse mask of the original dense model at the initialization, and then train only the sparse subnetwork from scratch, such as lottery ticket hypothesis (LTH) (Frankle & Carbin, 2018; Frankle et al., 2019), SNIP (Lee et al., 2018), GraSP (Wang et al., 2020), SynFlow (Tanaka et al., 2020), and even random pruning (Su et al., 2020; Frankle et al., 2020). In most (if not all) cases, sparse masks obtained by various those pruning algorithms are trained using the same training protocols optimized for dense neural network training. However, till now it is still *under-explored and unclear* if dense training protocols are optimal for training a sparse mask from scratch too. We hence ask: *Should training a sparse neural network requires atypical treatment and its own set of training toolkit or not?*

Orthogonally to the popular belief that quality of sparse masks matter the most for sparse training from scratch (good sparse masks train better), this paper explores an under-explored and alternative opportunity: *towards improving the*

[1]The University of Texas at Austin [2]University of California, Irvine. Correspondence to: Zhangyang Wang <atlaswang@utexas.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

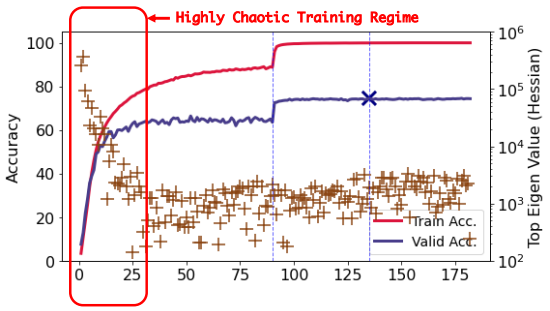

Figure 2. Top eigenvalues (Hessian) analysis of the training trajectory of a ResNet-18 sparse mask (90% sparsity) identified by LTH (Frankle & Carbin, 2018) using CIFAR-100.

training protocols of sparse sub-networks training. This paper demonstrate that one can carefully customize the sparse training protocols to deviate from the default dense network training protocols, and achieve significantly better performance from the same sparse mask. Our work is primarily motivated by the observation thet sparse training suffers from poor gradient flow (Tessera et al., 2021), and it has a highly chaotic optimization trajectory (Figure 2), which can lead to sub-optimal convergence of sparse masks. To tackle this problem, it is important to do a meticulous inspection of the training protocols of sparse training.

Our Contribution We offer a toolkit of *sparse retraining techniques* (ToST), and demonstrate that, solely by introducing our techniques into the training of sparse masks identified by existing popular pruning algorithms, our methods substantially reduce training instability, and improve performance and generalization of trained sub-networks. Our contributions can be summarized as:

- In contrary to the common belief that quality of masks matter the most for sparse retraining, we argue for an orthogonal, and under-explored theme: *improving the training techniques for pruned masks* and demonstrate that it significantly helps the sparse masks identified by various pruning algorithms to perform better.
- We provide a curated and easily adaptable *training toolkit (ToST)* for training ANY sparse mask from scratch: "ghost" skip-connection (injecting additional non-existent skip-connections in the sparse masks), "ghost" soft neurons (changing the ReLU neurons into smoother activation functions such as Swish (Ramachandran et al., 2017) and Mish (Misra, 2019)), as well as modifying initialization and labels.
- We report extensive experiments using variety of datasets, network architectures, and mask options. Incorporating our techniques in the sparse retraining immediately boosts the performance of sparse mask. This prompts to provide equal attention to improving the sparse retraining protocols rather than only focusing on designing better mask finding algorithms.

2. Methodology

In this section, we aim to provide a detailed introduction and motivation behind the tweaks in our sparse mask training toolkit (ToST). ToST consists of two **main tweaks**: "Ghost" Soft Neurons (GSw), and "Ghost" Skip Connections (Gsk), and some **miscellaneous tweaks** such as layer-wise re-scaled initialization, and label smoothing. We emphasize on highlighting the "Ghostliness" behaviour of GSw and GSk, i.e, how they can be temporarily incorporated in sparse mask without making any final architecture changes.

2.1. Revisiting Sparse Training

Given a dense network $f(\boldsymbol{\theta}, \cdot)$, the sparse sub-network of it is defined as $f(\boldsymbol{\theta} \odot \boldsymbol{m}, \cdot)$, where $\boldsymbol{m} \in \{0,1\}^{\|\boldsymbol{\theta}\|_0}$ is the binary mask indicating the sparsity levels, and $\odot$ is the element-wise product. $\boldsymbol{m}$ can be obtained from the pre-trained weights $\boldsymbol{\theta_d}$ or the random initialization $\boldsymbol{\theta_0}$. The sparse training aims to train $f(\boldsymbol{\theta} \odot \boldsymbol{m}, \cdot)$ from scratch with training protocols $\mathcal{P}$. Previous works mainly focus on how to find a better $\boldsymbol{m}$ with θ_0, while still apply the same $\mathcal{P}$ as the dense net $f(\boldsymbol{\theta}, \cdot)$.

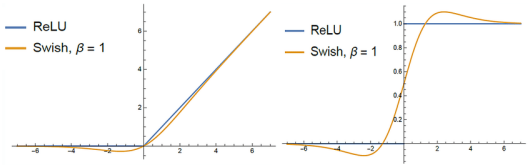

Figure 3. ReLU and Swish (Parametric Swish with $\beta = 1$) activation functions along with their derivatives.

2.2. "Ghost" Smooth Neurons and Skip Connection

Sparse Neural Networks and their trainability: Highly sparse networks easily suffer from the layer-collapse (Tanaka et al., 2020), i.e., the premature pruning of an entire layer. This could make the sparse network untrainable, as the gradient cannot be backpropagated through that layer. Additionally, in most (if not all) cases during the sparse neural network training, Rectified Linear Units (ReLU) (Nair & Hinton, 2010) are adapted as the default activation function ignoring the fact that ReLU is primarily optimized for training dense neural networks. However, the gradient of ReLU changes suddenly around zero (Figure 3), and this non-smooth nature of ReLU is an obstacle to sparse retraining because it leads to high activation sparsity into the subnetwork (with many pruned weights), likely blocking a healthy gradient flow (Table 1). During the training of sparse neural networks, we also observed that sparse training follows a highly chaotic optimization trajectory, which can lead to sub-optimal convergence of sparse subnetworks.

Activation	Layer 1	Layer 2	Layer 3	Layer 4
ReLU	27.14%	39.33%	39.48%	57.93%
Swish	0.31%	0.26%	0.24%	0.20%
Mish	1.09%	1.14%	1.03%	0.95%

Table 1. Layer-wise Activation sparsity of ResNet-18 sparse mask (90% sparsity) identified by LTH (Frankle & Carbin, 2018) and trained with CIFAR-100.

Injecting "GSw" and "GSk" in the sparse mask: The non-smooth behavior of ReLU leads to high activation sparsity in the sparse network, and decreases its trainability by blocking the gradient flow. To mitigate this issue and encourage healthier gradient flow, we propose to temporally replace the ReLU to Swish (Ramachandran et al., 2017) and Mish (Misra, 2019) during the training of sparse masks. Different from ReLU, Swish and Mish are both smooth non-monotonic activation functions. The non-monotonic property allows for the gradient of small negative inputs, which leads to a more stable gradient flow (Tessera et al., 2021) during the training.

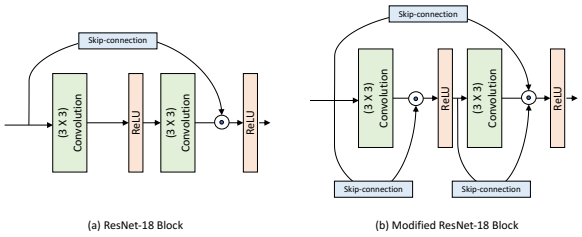

Figure 4. Our modified ResNet-18 block to introduce additional "ghost" skip-connections for the initial stage of sparse training.

With increase in sparsity, sparse neural networks suffers from layer-collapse (Tanaka et al., 2020) which leads to blockage of gradient flow during training. The skip connection (or named "residual-addition") (He et al., 2016) was initially proposed to avoid gradient vanishing problem, and enables the training of a very deep model. Motivated by the prevalent issue of dying kernels, high activation sparsity, and gradient blockage in highly sparse neural networks, we propose to "artificially" inject temporal new skip-connections during the training. Figure 4 illustrates this architectural modifications to the traditional Resnet-18 block. Similar to existing residual connection in traditional ResNet-18 block, our newly introduced skip-connections add input of each (3×3) convolution block, to their output before the activation. With high activation sparsity present in sparse subnetworks, additional skip-connections can facilitate healthy gradient flow and improve their trainability.

"Ghostliness" behaviour of "GSw" and "GSk": Deep neural networks have been identified to learn "low frequency features" initially (Rahaman et al., 2019) roughly before the first learning rate annealing, followed by the next stage of learning high frequency features (later part of training). Both GSk and GSw are considered to primarily help the first stage, because Swish compared to ReLU alleviates aliasing at zero-truncation via a smoother decay window, while residual connections add more DC components and smoothen the loss landscape (Li et al., 2017b). During the low frequency learning stage (initial part of training), incorporating GSk and GSw will help maintain healthy gradient flow while focusing on the learning the low-frequency features. Considering their limited role in the second stage, it provide an opportunity to gracefully remove them while the training progresses. Note that during our experiments we observe that either their abrupt removal, or being removed too late in the training, will hurt the optimization adversely and lead to poor generalization.

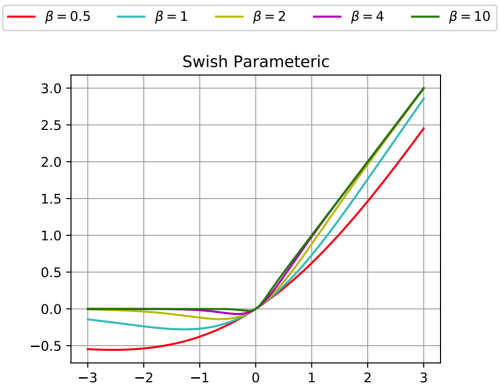

Figure 5. PSwish Visualization with different β values.

The parametric form of Swish (PSwish) function is $f(x) = x \times sigmoid(\beta x)$. Note that GSw is a special case of PSwish, when $\beta = 1$. PSwish transitions from identity function for $\beta = 0$, to ReLU for $\beta = \infty$ (smoothness decreases as β increases). In our effort to keep the sparse mask architecture unchanged, we gradually increase the β value of GSw, leading to be alike ReLU, and replace it with ReLU right before the first learning rate decay where the training regime changes (Leclerc & Madry, 2020). Following that, the training resumes as normal.

Similarly, for GSk, we introduced gate functions regulated by a hyperparameter α, which controls the contribution of GSk during the training. With $\alpha = 1$, GSk make full contribution during the training of the sparse mask. We decrease α in a scheduled way as training progresses, and finally set $\alpha = 0$, which completely removes the role of GSk, right before the first learning rate decay. Note that "Ghostliness" behaviour of GSw and GSk helps in reducing the additional training overhead, zeroing the inference overhead, and rehabilitating the original architecture of sparse mask.

One natural question which require attention following the

"Ghostliness" behaviour of GSw and GSk is: *If we keep GSw and GSk throughout sparse training until the end, how that will impact their performance?* During our experiments, we found that keeping GSw and GSk forever during training provides slightly better performance. However, it will change the original backbone structure (hence unfair), and either nonlinear neurons or denser skip connections will add additional hardware latency during inference. In practice, it is viewed as a design trade-off for sparse neural networks; yet in this paper we stick to the same backbone architecture (including both neuron and skip pattern) as provided.

2.3. Miscellaneous Tweaks

Layer-wise Re-scaled initialization (LRsI): Carefully crafted initializations that can prevent gradient explosion/vanishing in backpropagation have been important for the early success of feed-forward networks (He et al., 2016; Glorot & Bengio, 2010). Even with recent cleverly designed initialization rules, complex models with many layers and branches suffer from instability. For example, the Post-LN Transformer (Vaswani et al., 2017) can not converge without learning rate warmup using the default initialization.

In sparse subnetwork training, most existing works use common initializations (Glorot & Bengio, 2010; He et al., 2016) directly inherited from dense NN (with the sparse mask applied). In sparse masks, the number of incoming/outgoing connections is not identical for all the neurons in the layer (Evci et al., 2020b) and this raises direct concerns against the blind usage of dense network initialization for sparse subnetworks. Yet, (Evci et al., 2020b) also showed that completely random re-initialization of sparse subnetworks can lead the sparse masks to converge to poorer solutions.

To balance between these conflicting concerns, we propose to keep the original initialization of sparse masks intact for each parameter block and just re-scaled it by a learned scalar coefficient following recently proposed in (Zhu et al., 2021). Aware of the sensitivity and negative impact of changing initialization identified by (Evci et al., 2020c), we point that that linear scaling will **not** hurt the original sparse mask's initialization, thanks to the BatchNorm layer which will effectively absorb any linear scaling of the weights. More specifically, we optimized a small set of scalar coefficients to make the first update step (e.g., using SGD) as effective as possible at lowering the training loss. After the scalar coefficients are learned, the original initialization of sparse mask is re-scaled and the optimization proceeds as normal.

Label Smoothing (LS): Specifically, given the output probabilities p_k from the network and the target y_k, a network trained with hard labels aims to minimize the cross-entropy loss by $L_{\text{LS}} = -\sum_{k=1}^{K} y_k \log(p_k)$, where y_k is "1" for the correct class and "0" for others, and K is the number of classes. Label smoothing (Szegedy et al., 2016) changes the target to a mixture of hard labels with a uniform distribution, and minimizes the cross-entropy between the modified target $\frac{LS}{k}$ and output p_k. The modified target is defined as $y_k^{LS} = y_k(1-\alpha) + \alpha/K$, where α is the smooth ratio. This uniform distribution introduces smoothness into the training and encourages small logit gaps. Thus, label smoothing results in better model calibration and prevents overconfident predictions (Müller et al., 2019). In our work, we propose to incorporate label smoothing during the training of sparse masks and show that it can effectively help in improving the performance of sparse masks.

3. Experiments and Analysis

3.1. Settings

Following the recent developments in pruning algorithms, we have used sparse masks identified from various pruning techniques: LTH (Frankle & Carbin, 2018), SNIP (Lee et al., 2018), GraSP (Wang et al., 2020), SynFlow (Tanaka et al., 2020), as well as Random Pruning. Note that we have used the offical pytorch implementation of these algorithms to identify sparse mask, and train them with our ToST to evaluate the performance gain. For extensive validation across different datasets and architecture, we selected the most popular LTH (Frankle & Carbin, 2018), and show how ToST generalizes across different datasets and architecture.

In our experiments, all of our sparse masks has been trained using similar settings for simplicity in reproducing our results. For training, we adopt an SGD optimizer with momentum 0.9 and weight decay $2\text{e}{-4}$. The initial learning rate is set to 0.1, and the networks are trained for 180 epochs with a batch size of 128. The learning rate decays by a factor of 10 at the 90th and 135th epoch during the training. We run all our experiments 3 times to obtain more stable and reliable test accuracies. Note that apart from the tweaks proposed in ToST, we make no additional modification during the training process for fair evaluation.

3.2. ToST and ANY MASK

In this section, we conduct a systematic study to understand the performance gain by our proposed sparse training toolkit (ToST), when they are incorprated in the training process of ANY sparse identified by various pruning algorithms. Table 2 demonstrate the effectiveness of our proposed toolkit on the sparse masks obtained by recently proposed pruning algorithms: SNIP (Lee et al., 2018), which is a sensitivity based pruning algorithm, GraSP (Wang et al., 2020), which is a Hessian based pruning algorithm, SynFlow (Tanaka et al., 2020), which is an iterative data-agnostic pruning algorithm, Lottery Ticket (Frankle & Carbin, 2018), which is based on iterative magnitude pruning, as well as Random Pruning . We have used CIFAR-10 and CIFAR-100 during the evaluation of our ToST on various sparse masks obtained by pruning a substantial amount of param-

Sparse Mask	CIFAR-10			CIFAR-100		
	90%	95%	98%	90%	95%	98%
ResNet-32 [No Pruning]	94.80	-	-	74.64	-	-
Random Pruning	89.95±0.23	89.68±0.15	86.13±0.25	63.13±2.94	64.55±0.32	19.83±3.21
Random Pruning + ToST	**91.53±0.11**	**91.44±1.01**	**88.20±0.89**	**65.19±1.36**	**64.61±1.21**	**33.98±6.64**
SNIP (Lee et al., 2018)	92.26±0.32	91.18±0.17	87.78±0.16	69.31±0.52	65.63±0.15	55.70±1.13
SNIP + ToST	**92.83±0.15**	**92.01±0.21**	**88.12±0.13**	**70.00±0.09**	**68.46±0.62**	**60.21±1.96**
GraSP (Wang et al., 2020)	92.20±0.31	91.39±0.25	88.70±0.42	69.24±0.24	66.50±0.11	58.43±0.43
GraSP + ToST	**92.98±0.07**	**92.77±0.14**	**89.92±0.56**	**70.18±0.22**	**67.20±0.74**	**62.30±1.06**
SynFlow (Tanaka et al., 2020)	92.01±0.22	91.67±0.17	88.10±0.25	69.03±0.20	65.23±0.31	58.73±0.30
SynFlow + ToST	**93.39±0.59**	**92.06±0.32**	**91.82±0.73**	**70.25±0.06**	**67.90±1.22**	**61.72±0.84**
LTH (Frankle & Carbin, 2018)	93.14±0.30	92.98±0.12	92.22±0.61	71.11±0.57	70.37±0.19	69.02±0.22
LTH + ToST	**94.01±0.23**	**93.60±0.70**	**93.34±1.06**	**72.30±0.61**	**71.99±0.95**	**70.22±0.61**
ResNet-50 [No Pruning]	94.90	-	-	74.91	-	-
Random Pruning	85.11±4.51	88.76±0.21	85.32±0.47	65.67±0.57	60.23±2.21	28.32±10.35
Random Pruning + ToST	**92.73±0.22**	**90.95±1.22**	**87.11±2.21**	**67.75±1.32**	**63.60±0.11**	**41.99±4.51**
SNIP (Lee et al., 2018)	91.95±0.13	92.12±0.34	89.26±0.23	70.43±0.43	67.85±1.02	60.38±0.78
SNIP + ToST	**92.89±0.53**	**92.56±0.12**	**90.56±0.19**	**70.79±0.22**	**68.06±0.09**	**61.51±1.41**
GraSP (Wang et al., 2020)	92.10±0.21	91.74±0.35	89.97±0.25	70.53±0.32	67.84±0.25	63.88±0.45
GraSP + ToST	**92.64±0.17**	**92.33±0.09**	**90.94±0.35**	**70.89±0.21**	**68.09±0.12**	**65.01±0.33**
SynFlow (Tanaka et al., 2020)	92.05±0.20	91.83±0.23	89.61±0.17	70.43±0.30	67.95±0.22	63.95±0.11
SynFlow +ToST	**92.55±0.10**	**92.57±0.18**	**90.27±0.29**	**70.86±0.21**	**68.83±0.15**	**65.40±0.13**
LTH (Frankle & Carbin, 2018)	93.69±0.31	93.18±0.17	92.79±0.14	71.89±0.11	71.05±0.13	70.41±0.28
LTH + ToST	**94.37±0.06**	**94.01±0.32**	**92.94±0.21**	**73.69±0.13**	**72.20±0.15**	**71.93±0.34**

Table 2. Classification accuracies of various pruning algorithm for varying sparsities $s \in \{90\%, 95\%, 98\%\}$ and network architectures (ResNet-18 and 32) with and without our sparse training toolkit (ToST).

eters of ResNet-18 and ResNet-50 with varying sparsities $s \in \{90\%, 95\%, 98\%\}$.

The results are summarized in Table 2. We first observe that among all the pruning methods, sparse masks obtainned by LTH (Frankle & Carbin, 2018) perform the best at high sparsity for both CIFAR-10 and CIFAR-100. In comparison, sparse mask trained with ToST stays stable in performing significantly better across all pruning methods, datasets, and network architectures. Very interestingly, we observe that ToST can help randomly pruned masks at 98% sparsity to achieve up to $\sim 14\%$ and $\sim 13\%$ (CIFAR-100) better results, for ResNet-32 and ResNet-50 respectively. This provided a strong indication towards the training stability provided by ToST during the sparse training even the mask quality is not great. Similarly for GraSP mask with 98% sparsity, ToST provides $\sim 4\%$ improvement.

We additionally evaluated SNIP and LTH sparse masks with sparsities $s \in \{85\%, 90\%, 95\%\}$ on **TinyImageNet** (Deng et al., 2009). Table 3 presents the summary of our results. Similar to our results on CIFAR-10 and CIFAR-100, ToST provided sufficient performance boost to ResNet-50 sparse masks identified by SNIP and LTH, on the larger TinyImageNet dataset. At 95% sparsity, it provides $> 2\%$ improvement for SNIP, and $> 1.5\%$ improvement for the LTH mask. These benefits prompt a greater potential to reconsider the exploration and provide attention to improving sparse retraining strategies.

3.3. Performance Breakdown of ToST

Our toolkit (ToST) consists of two **main tweaks**: "Ghost" Soft Neurons (GSw), and "Ghost" Skip Connections (Gsk), and some **miscellaneous tweaks** such as layer-wise rescaled initialization, and label smoothing. While these tweaks when jointly applied in the training of sparse masks, significantly provides huge performance gain (Table 2, 3), an obvious question is: *How our proposed tweaks helps in performance of sparse masks, when they are applied in isolation?* To answer this question, we selected LTH masks (considering it high popularity and better performance at high sparsity) with varying sparsities $s \in \{75\%, 80\%, 85\%, 90\%, 95\%\}$ for detailed evaluation of our individual tweaks.

Table 4 summarizes the performance comparison of our individual tweaks when they are applied in isolation during the training of sparse masks. We can observe that "GSk" standalone is the most effective tweak in improving the performance at very high sparsity with a performance gain of 2.15% at 95% sparsity. "GSw" stands out to be the second most effective tweak in our toolkit for high sparsity, with performance gain of 1.36% and 1.28% at sparsity level 85% and 95% respectively. It is worth noticing that "LRsI" achieve highest performance gain at 75% sparsity which hints that each tweak helps in the trainability of sparse networks in its own unique way. When we combine these tweaks together to train the sparse LTH tickets, we observe

Algorithm	85%	90%	95%
SNIP (Lee et al., 2018)	58.91±0.23	56.15±0.31	51.19±0.47
SNIP + ToST	59.44±0.09	57.19±0.21	53.21±0.08
LTH (Frankle & Carbin, 2018)	60.11±0.13	58.46±0.17	53.19±0.31
LTH + ToST	61.52±0.32	58.96±0.08	54.76±0.22

Table 3. Classification accuracies on TinyImageNet for varying sparsities $s \in \{90\%, 95\%, 98\%\}$ using ResNet-50.

Method	75%	80%	85%	90%	95%
LTH (Frankle & Carbin, 2018)	73.21±0.17	72.94±0.12	71.91±0.22	71.12±0.30	69.57±0.19
LTH + GSk	73.77±0.11	**73.69±0.25**	72.86±0.30	**72.17±0.23**	**71.72±0.22**
LTH + GSw	73.45±0.13	73.22±0.43	**73.27±0.31**	72.03±0.12	70.85±0.52
LTH + LRsI	**73.93±0.15**	73.12±0.13	72.30±0.19	71.83±0.32	69.98±0.29
LTH + LS	73.58±0.28	73.70±0.32	72.65±0.25	71.93±0.20	70.19±0.14
LTH + ToST	74.29±0.31	74.03±0.14	73.90±0.49	73.23±0.27	72.08±0.10

Table 4. Breakdown of the performance of individual tweaks in ToST tweaks when applied on training ResNet-18 sparse masks (LTH) with varying sparsities $s \in \{75\%, 80\%, 85\%, 90\%, 95\%\}$ and trained on CIFAR-100.

the overall performance boost is significantly better than the individual tweaks. We get $1.08\% - 2.51\%$ performance gain within the sparsity range of $s \in [75 - 95]\%$. This clearly highlights the orthogonal benefits of our tweaks in sparse mask training.

3.4. "Ghostliness" of GSw and GSk

As discussed in Section 2.2, "GSk" and "GSw" primarily help in first stage of learning, we are motivated to remove them gradually during the course of time (aka. "ghostliness"). Gradual removal will help in reducing the additional training overhead, zeroing the inference overhead, and rehabilitating the original architecture of sparse mask. To investigate the impact of "ghostliness" behaviour, and how it may impact in unleashing the true strength of "GSk" and "GSw", when they are kept throughout sparse training until the end, we attempted to compare the performance with and without "ghostliness" of our tweaks.

Figure 6 summarizes the performance comparison of the "Ghostiliness" behaviour of GSk and GSw with the default prolonged injection of swish and skip connections for LTH sparse masks with varying sparsities $s \in \{80\%, 85\%, 90\%, 95\%\}$. We observed that keeping the skip connections, and swish throughout sparse training until the end provides some additional performance benefit (marginal for swish), but it comes up at the cost of additional hardware latency during the inference time. In practice, we identify this as a design trade-off for the sparse neural networks. To complete the analysis, we attempted to analyse mask performance if we ghost GSk and GSw after the first learning rate decay, and we found that the performance decreases by -0.917% and -0.429% (95% sparsity) for GSk and GSw respectively, compare to our proposed settings. Additionally, abrupt removal of GSk and GSw towards the end of training, leads to significant performance drop of $> 1.2\%$ (sparsity 95%) for both GSk and GSw.

4. Ablation and Analysis

4.1. Generalization across Datasets and Architectures

In this section, we additionally evaluate the performance of ToST on VGG-16 (Simonyan & Zisserman, 2014) and MobileNet (Howard et al., 2017) using CIFAR-10 and CIFAR-100. Note that we have mainly studied LTH (Frankle & Carbin, 2018) masks hereinafter for ablations, considering their superior performance in comparison to SNIP, SynFlow, GraSP, and Random Pruning. Figure 7 summarizes the performance comparison of our sparse training toolkit when it is used to train the LTH sparse tickets with sparsity ranging from $s \in [20\% - 97\%]$. In the plot, the red line indicates the performance of sparse tickets when they are trained using default setting proposed in (Frankle & Carbin, 2018; Frankle et al., 2019) while the blue line indicates the sparse tickets when trained using our ToST without any other additional modification for fair comparison. Clearly, our proposed tweaks help significantly in improving the performance of sparse tickets across all sparsities. Moreover, it is important to observe that the performance benefits of our tweaks increases significantly with increase in the sparsity level. This observation augment the necessity of ToST, while training sparse subnetworks with high sparsity.

4.2. Smoothness of Loss Landscape

In this section, we try to understand the implications of our techniques during training of sparse subnetworks, through

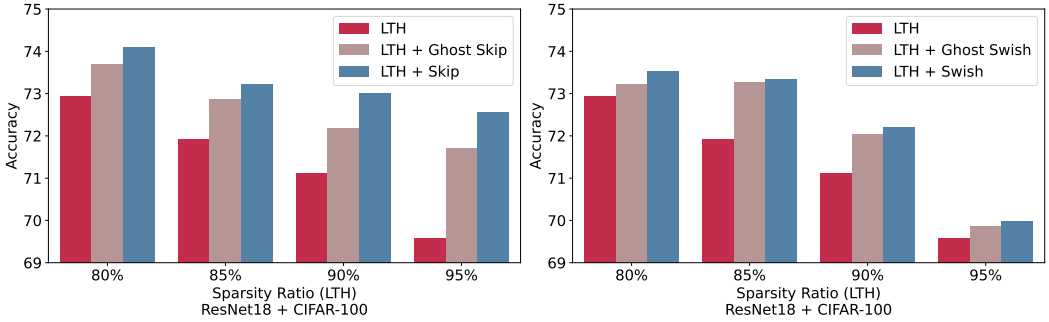

Figure 6. Performance comparison of the "Ghostiliness" behaviour of GSk and GSw with the default prolonged injection of swish and skip connections for LTH sparse masks with varying sparsities $s \in \{80\%, 85\%, 90\%, 95\%\}$.

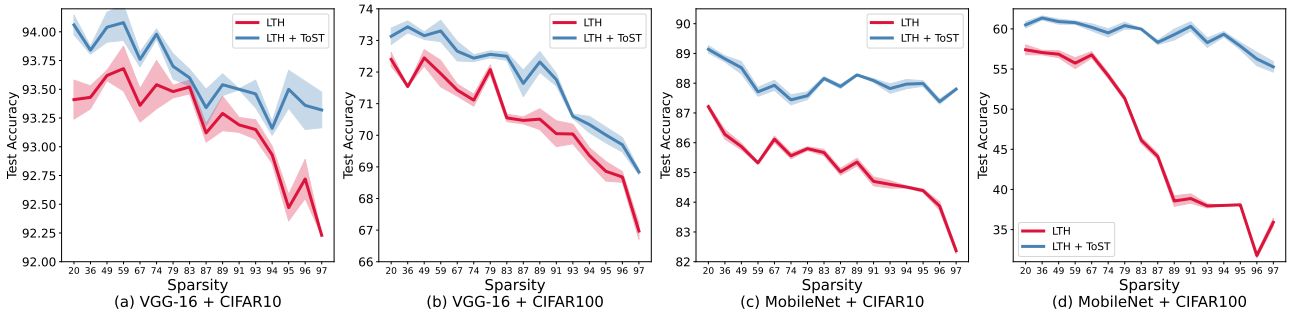

Figure 7. Performance comparison of sparse masks by LTH at varying sparsities $s \in [20\% - 97\%]$ on CIFAR-10 and CIFAR-100.

some common lens. Our methods can be viewed as a form of *learned smoothening* (Chen et al., 2021b) which is incorporated at an early training stage. Smoothening tools can be applied on the logits (naive label-smoothening (Müller et al., 2019), knowledge distillation (Hinton et al., 2015a)), on the weight dynamics (stochastic weight averaging (Izmailov et al., 2018)), or on regularizing the end solution.

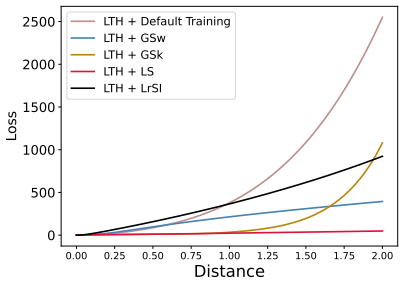

Figure 8. The change in testing loss as a function of perturbed weight distance, in the direction of top eigenvector of Hessian matrix (Yao et al., 2020) of LTH ticket (90% sparsity) for ResNet-18 trained on CIFAR-100.

We expect tweaks in ToST to find flatter minima for sparse mask training to improve its generalization, and we show it to indeed happen by visualizing the loss landscape w.r.t both input and weight spaces. Figure 10 shows the comparison of loss landscape of LTH ticket (90% sparsity) from Resnet-18 trained using default dense training protocols proposed in (Frankle & Carbin, 2018; Frankle et al., 2019) and individual tweaks in our sparse toolkit ToST. It can be observed that each one of our tweaks notably flatten the sharp landscape w.r.t. the input space, compare to the default baseline of using (Frankle & Carbin, 2018; Frankle et al., 2019), which aligns with our hypothesis that our tweaks can be viewed as some form of "learned smoothening".

Figure 8 follows (Yao et al., 2020) to perturb the trained LTH sparse mask (90% sparsity) in weight space, to show the flattening effect of our tweaks. It shows the change in testing loss as a function of perturbed weight space, in the direction of top eigenvector of Hessian obtained by (Yao et al., 2020). Our methods present better weight smoothness around the achieved local minima, which suggests improved generalization (Dinh et al., 2017; Petzka et al., 2019).

	Dense NN (0%)	**20%**	**75%**	**95%**
"GSk"	-0.77%	+0.03%	+0.56%	+2.15%
"GSw"	+0.11%	+0.29%	+0.24%	+1.28%

Table 5. Performance benefit of "GSk" and "GSW" when applied to dense networks (0%) sparsity, low sparsity (20%), mid-level sparsity (75%), and high sparsity (95%). We have used LTH sparse mask of ResNet-18 trained on CIFAR-100.

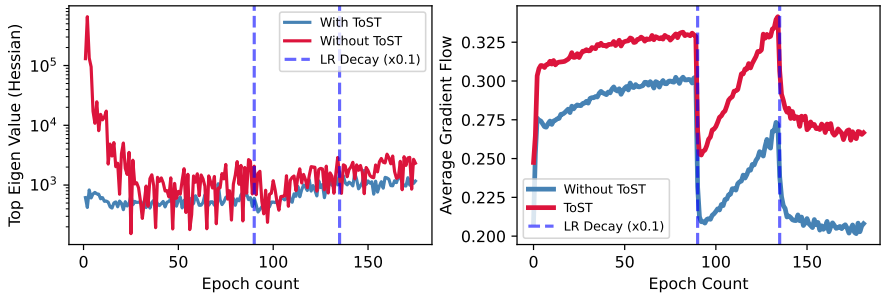

Figure 9. (a) Comparison of Top eigenvalues (Hessian) of training trajectory of a ResNet-18 sparse mask (90% sparsity by LTH) on CIFAR-100 with and without ToST. (b) Comparison of Average Gradient Flow (Tessera et al., 2021) ResNet-18 sparse mask (90% sparsity by LTH) on CIFAR-100 during training with and without ToST.

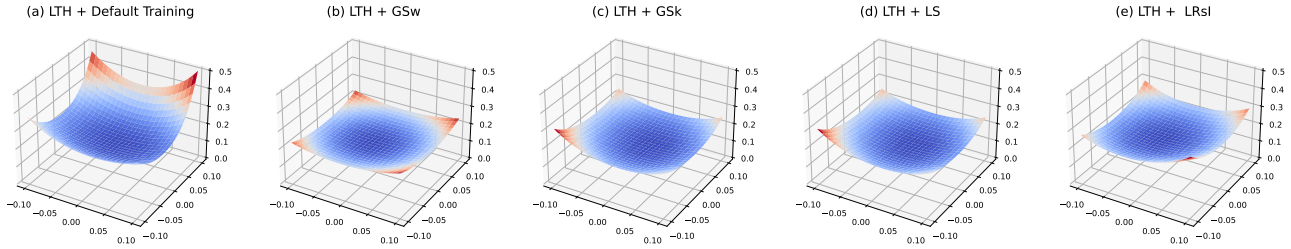

Figure 10. Comparison of loss landscape of LTH ticket (90% sparsity) from Resnet-18 trained using default dense training protocols proposed in (Frankle & Carbin, 2018; Frankle et al., 2019) and individual tweaks in our sparse toolkit ToST. Loss plots are generated with the same original images randomly chosen from CIFAR-100 test dataset using (Li et al., 2017a). z-axis denote the loss value.

4.3. Are "GSk" and "GSw" same helpful in Dense NNs?

In this section we attempt to answer one important question: *How does "GSw" and "GSk" impact the performance of dense network? Are they equally beneficial to training dense networks too?* Table 5 illustrates the performance benefits of GSw and GSk when they are applied at various level of sparsity ranging from 0% (corresponds to dense network) to low-level (20%), mid-level (75%), and finally high-level (95%). It clearly answer aforementioned question that both "GSk" and "GSw" significantly help the sparse networks more than the dense network and the performance benefits enlarges with increasing sparsity.

Remark: The recently proposed RepVGG (Ding et al., 2021) cleverly adds skip-connections (SKs) to dense networks of *VGG-like plain topology* (no SKs) by re-parameterization during training, and later removing SKs at inference. In contrast, our "GSk" is applied to training sparse networks of *arbitrary topology*, mostly ResNets with pre-existing native SKs. Our experiments further reveal that adding extra SKs can even **hurt** the performance of dense ResNets with pre-existing SKs (e.g., "-0.77%" in Table 5 for ResNet-18). Meanwhile, adding SKs during sparse training of those ResNets, using our proposed soft alternative, benefits their performance consistently, especially at high sparsity. *Our lesson is*: sparsity has an overlooked important role in influencing whether more skip connections will benefit, potentially due to the trade-off between network representation capacity and optimization easiness.

4.4. Effect of ToST on Hessian and Gradient Flow

Hessian eigenvalue/spectral density (Yao et al., 2020) can be used to analyze the the topology of the loss landscape, and its magnitude indicates the "degree of smoothness in loss landscape" and the ease for Stochastic Gradient Descent to converge to a good solution. Higher value of top eigenvalue indicate poorer quality of loss landscape and difficult optimization. Figure 9(a) illustrates the effect of ToST on top eigenvalues of Hessian for the training trajectory of ResNet-18 sparse LTH mask with 90% sparsity. Furthermore, to effectively measure the gradient changes before and after ToST, we calculated the *Average Gradient Flow* for the unpruned weights during training (Tessera et al., 2021). Figure 9(b) presents the comparison of the gradient flow with and without ToST when training ResNet-18 sparse LTH mask with 90% sparsity. Clearly, ToST facilitates healthier gradient flow during the training of sparse neural networks.

5. Related Work

Network Pruning Pruning is fruitful in reducing network inference costs. In general, there are two types of pruning: One is unstructured pruning, which usually removes

redundant weights. The important score of weights can be obtained from magnitude (Han et al., 2015a;b), gradient (Molchanov et al., 2017; 2019) or Hessian (LeCun et al., 1990). The other is structured pruning, which prunes the entire channels or layers (Liu et al., 2017; Li et al., 2016; Wen et al., 2016; He et al., 2017). All of them starts with the fully trained dense model, and finetune the sparse network to achieve similar accuracy.

Sparse Training Sparse training aims to train a sparse network from scratch. It can be categorized into two groups: ① *Static sparse training*, which prunes the network at the initialization and maintains the pruning mask throughout training. Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2018; Frankle et al., 2019; Evci et al., 2019; Savarese et al., 2020; Chen et al., 2020a; Gale et al., 2019; Chen et al., 2020b) suggests that a dense network contains several sparse sub-network that can match the accuracy of the original model when trained in isolation from scratch. Later on, the Single-Shot Network Pruning (SNIP) (Lee et al., 2018) uses the gradients of the training loss at initialization to prune the network. The Gradient Signal Preservation (GraSP) (Wang et al., 2020) prune connections based on the gradient flow. The Iterative Synaptic Flow Pruning (SynFlow) (Tanaka et al., 2020) preserves the total flow of synaptic strengths through the network to handle the layer-collapse issue. (Sung et al., 2021) selects the sparse mask via Fisher information. ② *Dynamic sparse training*, which allows the pruning mask to be updated during the training. They usually prunes weights based on the magnitude and grows weights back (Mocanu et al., 2018) at random or based on the gradient (Evci et al., 2020a; Liu et al., 2021; Chen et al., 2022; 2021a). All of these works imply that the quality of pruning mask is vital in sparse training.

(Lee et al., 2019) analyzed sparse subnetworks from the signal propagation perspective and proposes a new technique of re-fitting initialization to improve their trainablity. More specifically, provided with sparse topology C and initial random weights W, (Lee et al., 2019) optimizes $W \rightarrow W^*$ such that the combination of the sparse topology and weights become layerwise orthogonal. In comparison, our LRsI technique keeps the original sparse weight initialization, except learning a small set of scaling coefficients per block to improve the gradient propagation. Along with being computationally more efficient, it can also better preserve the *good initialization* already found in some sparse mask schemes such as LTH, which have been confirmed as necessary for their success (Tessera et al., 2021).

Smoothness in Neural Network Infusing smoothness into neural networks, including on the weights, logits, or training trajectory, is a common techniques to improve the generalization and optimization (Jean & Wang, 1994). For labels, smoothness is usually introduced by replacing the hard target with soft labels (Szegedy et al., 2016) or soft logits (Hinton et al., 2015a). This uncertainty of labels helps to alleviate the overconfidence and improves the generalization. Smoothness can also implemented by replacing the activation functions (Misra, 2019; Ramachandran et al., 2017), adding skip-connections in NNs (He et al., 2016), or averaging along the trajectory of gradient descent (Izmailov et al., 2018). These methods contribute to more stable gradient flows (Tessera et al., 2021) and smoother loss landscapes, but most of them have not been considered nor validated on sparse NNs.

6. Conclusion

This paper takes one step towards improving the training techniques for sparse neural networks. Contrary to the popular belief that only the quality of sparse masks matters for sparse training, this paper presents an alternative opportunity that one can carefully customize the sparse training techniques to train sparse sub-networks identified by various pruning algorithms, and achieve significant performance benefits. It presents a curated and easily adaptable training toolkit for training any sparse mask from scratch, without any additional overhead. Extensive experiments across different pruning algorithms, sparse masks, and datasets shows the effectiveness of the proposed toolkit. Our future work will aim for more theoretical understanding of the role of our toolkit in sparse training performance improvement.

Acknowledgement

Z.W. is in part supported by an NSF RTML project (#2053279).

References

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

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. The lottery ticket hypothesis for pre-trained bert networks. *arXiv*, abs/2007.12223, 2020b.

Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., and Wang, Z. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34:19974–19988, 2021a.

Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. Robust overfitting may be mitigated by properly learned smoothening. In *International Conference on Learning Representations*, 2021b.

Chen, T., Zhang, Z., pengjun wang, Balachandra, S., Ma, H., Wang, Z., and Wang, Z. Sparsity winning twice: Better robust generalization from more efficient training. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=SYuJXrXq8tw.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., and Sun, J. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13733–13742, 2021.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *ICML*, 2017.

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*, pp. 2943–2952. PMLR, 2020a.

Evci, U., Ioannou, Y. A., Keskin, C., and Dauphin, Y. Gradient flow in sparse neural networks and how lottery tickets win. *ArXiv*, abs/2010.03533, 2020b.

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*, 2020c.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

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*, 2020.

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

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

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, 2015b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015a.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015b.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2018. URL http://jmlr.org/papers/v18/16-456.html.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Jean, J. S. and Wang, J. Weight smoothing to improve network generalization. *IEEE Transactions on neural networks*, 5(5):752–763, 1994.

Leclerc, G. and Madry, A. The two regimes of deep network training. *arXiv preprint arXiv:2002.10376*, 2020.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *NIPS*, 1989.

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. H. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

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

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

Li, H., Xu, Z., Taylor, G., and Goldstein, T. Visualizing the loss landscape of neural nets. *CoRR*, abs/1712.09913, 2017a. URL http://arxiv.org/abs/1712.09913.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017b.

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

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pp. 2736–2744, 2017.

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

Misra, D. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 4: 2, 2019.

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*, pp. 2498–2507. PMLR, 2017.

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.

Müller, R., Kornblith, S., and Hinton, G. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? *CoRR*, abs/1906.02629, 2019. URL http://arxiv.org/abs/1906.02629.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Petzka, H., Adilova, L., Kamp, M., and Sminchisescu, C. A reparameterization-invariant flatness measure for deep neural networks. *ArXiv*, abs/1912.00058, 2019.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Savarese, P., Silva, H., and Maire, M. Winning the lottery with continuous sparsification. In *NeurIPS*, 2020.

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

Su, J., Chen, Y., Cai, T., Wu, T., Gao, R., Wang, L., and Lee, J. D. Sanity-checking pruning methods: Random tickets can win the jackpot. *arXiv preprint arXiv:2009.11094*, 2020.

Sung, Y.-L., Nair, V., and Raffel, C. Training neural networks with fixed sparse masks. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Uwh-v1HSw-x.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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.

Tessera, K.-a., Hooker, S., and Rosman, B. Keep the gradients flowing: Using gradient flow to study sparse network optimization. *arXiv preprint arXiv:2102.01670*, 2021.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020.

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.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. *2020 IEEE International Conference on Big Data (Big Data)*, pp. 581–590, 2020.

Zhu, C., Ni, R., Xu, Z., Kong, K., Huang, W. R., and Goldstein, T. Gradinit: Learning to initialize neural networks for stable and efficient training. *ArXiv*, abs/2102.08098, 2021.