The Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)

Gradient Flow in Sparse Neural Networks and How Lottery Tickets Win

Utku Evci[*1], **Yani Ioannou**[*2], **Cem Keskin**[3], **Yann Dauphin**[1]

[1] Google, [2] University of Calgary, [3] Facebook
evcu@google.com, yani.ioannou@ucalgary.ca, cem.keskin@fb.com, ynd@google.com

Abstract

Sparse Neural Networks (NNs) can match the generalization of dense NNs using a fraction of the compute/storage for inference, and have the potential to enable efficient training. However, naively training unstructured sparse NNs from random initialization results in significantly worse generalization, with the notable exceptions of Lottery Tickets (LTs) and Dynamic Sparse Training (DST). Through our analysis of gradient flow during training we attempt to answer: (1) why training unstructured sparse networks from random initialization performs poorly and; (2) what makes LTs and DST the exceptions? We show that sparse NNs have poor gradient flow *at initialization* and demonstrate the importance of using sparsity-aware initialization. Furthermore, we find that DST methods significantly improve gradient flow *during training* over traditional sparse training methods. Finally, we show that LTs do not improve gradient flow, rather their success lies in re-learning the pruning solution they are derived from — however, this comes at the cost of learning novel solutions.

1 Introduction

Deep Neural Networks (DNNs) are the state-of-the-art method for solving problems in computer vision, speech recognition, and many other fields. While early research in deep learning focused on application to new problems, or pushing state-of-the-art performance with ever larger/more computationally expensive models, a broader focus has emerged towards their efficient real-world application. One such focus is on the observation that only a sparse subset of this dense connectivity is required for inference, as apparent in the success of *pruning*.

Pruning has a long history in Neural Network (NN) literature (Mozer and Smolensky 1989; Han et al. 2015), and remains the most popular approach for finding sparse NNs. Sparse NNs found by pruning algorithms (Han et al. 2015; Zhu and Gupta 2018; Molchanov, Ashukha, and Vetrov 2017; Louizos, Ullrich, and Welling 2017) (i.e. *pruning solutions*) can match dense NN generalization with much better efficiency at *inference* time. However, naively *training* an (unstructured) sparse NN from a random initialization (i.e. *from scratch*), typically leads to significantly worse generalization.

Two methods in particular have shown some success at addressing this problem — Lottery Tickets (LTs) and Dynamic Sparse Training (DST). However, we don't know how to find Lottery Tickets (LTs) efficiently; while RigL (Evci et al. 2020), a recent DST method, requires 5× the training steps to match dense NN generalization. Only in understanding how these methods overcome the difficulty of sparse training can we improve upon them.

A significant breakthrough in training dense DNNs — addressing vanishing and exploding gradients — arose from understanding gradient flow both at initialization, and during training. In this work we investigate the role of gradient flow in the difficulty of training unstructured sparse NNs from random initializations and from LT initializations. Our experimental investigation results in the following insights:

1. **Sparse NNs have poor gradient flow at initialization.** In §3.1, §4.1 we show that the predominant method for initializing sparse NNs is incorrect in not considering heterogeneous connectivity. We believe we are the first to show that sparsity-aware initialization methods improve gradient flow and training.
2. **Sparse NNs have poor gradient flow during training.** In §3.2, §4.2, we observe that even in sparse NN architectures less sensitive to incorrect initialization, the gradient flow *during training* is poor. We show that DST methods achieving the best generalization have improved gradient flow, especially in early training.
3. **Lottery Tickets don't improve upon (1) or (2), instead they re-learn the pruning solution.** In §3.3, §4.3 we show that a LT initialization resides within the same basin of attraction as the original pruning solution it is derived from, and a LT solution is highly similar to the pruning solution in function space.

2 Related Work

Pruning Pruning is used commonly in NN literature to obtain sparse networks (Mozer and Smolensky 1989; Han et al. 2015; Kusupati et al. 2020). While the majority of pruning algorithms focus on pruning *after* training, a subset focuses on pruning NNs *before* training (Lee, Ajanthan, and Torr 2019; Wang, Zhang, and Grosse 2020; Tanaka et al. 2020). Gradient Signal Preservation (GRaSP) (Wang, Zhang, and Grosse 2020) is particularly relevant to our study, since

[*] These authors contributed equally.

their pruning criteria aims to preserve gradient flow, and they observe a positive correlation between gradient flow *at initialization* and final generalization. However, the recent work of Frankle et al. (2020b) suggests that the reported gains are due to sparsity distributions discovered rather than the particular sub-network identified. Another limitation of these algorithms is that they don't scale to large-scale tasks like ResNet-50 training on ImageNet-2012.

Lottery Tickets Frankle and Carbin (2019) showed the existence of sparse sub-networks at initialization — known as Lottery Tickets — which can be trained to match the generalization of the corresponding dense DNN. The initial work of Frankle and Carbin (2019) inspired much follow-up work. Liu et al. (2019); Gale, Elsen, and Hooker (2019) observed that the initial formulation was not applicable to larger networks with higher learning rates. Frankle et al. (2019, 2020a) proposed *late rewinding* as a solution. Morcos et al. (2019); Sabatelli, Kestemont, and Geurts (2020) showed that LTs trained on large datasets transfer to smaller ones, but not *vice versa*. Zhou et al. (2019); Frankle, Schwab, and Morcos (2020); Ramanujan et al. (2019) focused on further understanding LTs, and finding sparse sub-networks at initialization. However, it is an open question whether finding such networks at initialization could be done more efficiently than with existing pruning algorithms.

Dynamic Sparse Training Most training algorithms work on pre-determined architectures and optimize parameters using fixed learning schedules. Dynamic Sparse Training (DST), on the other hand, aims to optimize the sparse NN connectivity jointly with model parameters. Mocanu et al. (2018); Mostafa and Wang (2019) propose replacing low magnitude parameters with random connections and report improved generalization. Dettmers and Zettlemoyer (2019) proposed using momentum values, whereas Evci et al. (2020) used gradient estimates directly to guide the selection of new connections, reporting results that are on par with pruning algorithms, and has been applied to vision transformers (Chen et al. 2021), language models (Dietrich et al. 2021), reinforcement learning (Sokar et al. 2021), training recurrent neural networks (Liu et al. 2021b) and fast ensembles (Liu et al. 2021a). In §4.2 we study these algorithms and try to understand the role of gradient flow in their success.

Random Initialization of Sparse NN In training sparse NN from scratch, the vast majority of pre-existing work on training sparse NN has used the common initialization methods (Glorot and Bengio 2010; He et al. 2015) derived for *dense* NNs, with only a few notable exceptions. Liu et al. (2019); Ramanujan et al. (2019); Gale, Elsen, and Hooker (2019) scaled the variance (fan-in/fan-out) of a sparse NN layer according to the layer's sparsity, effectively using the standard initialization for a small dense layer with an equivalent number of weights as in the sparse model. Lee et al. (2020) measures the singular values of the input gradients and proposes to use a dense orthogonal initialization to ensure dynamical isometry and improve one-shot pruning performance before training. Similar to our work, Tessera, Hooker, and Rosman (2021); Lubana and Dick (2021) compares dense and sparse networks at initialization and during training using gradient flow, whereas Golubeva, Neyshabur, and Gur-Ari (2021) study the distance to the infinite-width kernel.

3 Analyzing Gradient Flow in Sparse NNs

A significant breakthrough in training very deep NNs arose in addressing the *vanishing/exploding gradient* problem, both at initialization, and during training. This problem was understood by analyzing the signal propagation within a DNN, and addressed in improved initialization methods (Glorot and Bengio 2010; He et al. 2015; Xiao et al. 2018) alongside normalization methods, such as Batch Normalization (BatchNorm) (Ioffe and Szegedy 2015). In our work, similar to Wang, Zhang, and Grosse (2020), we study these problems using gradient flow, $\nabla L(\theta)^T \nabla L(\theta)$ which is the first order approximation* of the decrease in the loss expected after a gradient step. We observe poor gradient flow for the predominant sparse NN initialization strategy and propose a sparsity-aware generalization in §3.1. Then in §3.2 and §3.3 we summarize our analysis of DST methods and the LT hypothesis respectively.

3.1 The Initialization Problem in Sparse NNs

Here we analyze the gradient flow at initialization for random sparse NNs, motivating the derivation of a more general initialization for NN with heterogeneous connectivity, such as in unstructured sparse NNs.

In practice, without a method such as BatchNorm (Ioffe and Szegedy 2015), using the correct initialization can be the difference between being able to train a DNN, or not — as observed for VGG16 in our results (§4.1, Table 1). The initializations proposed by Glorot and Bengio (2010); He et al. (2015) ensure that the output distribution of every neuron in a layer is zero-mean and of unit variance by sampling initial weights from a Gaussian distribution with a variance based on the number of incoming/outgoing connections for all the neurons in a dense layer, as illustrated in Fig. 1a, which is assumed to be identical for all neurons in the layer.

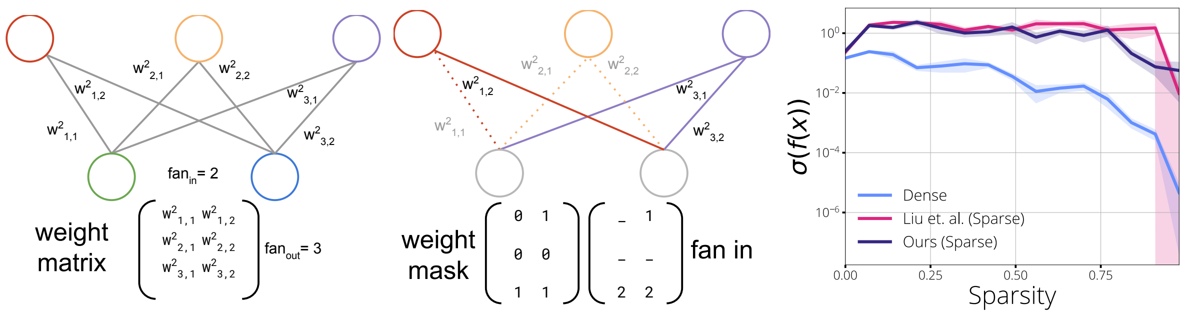
In an unstructured sparse NN however, the number of incoming/outgoing connections is not identical for all neurons in a layer, as illustrated in Fig. 1b. Here we will focus only on explaining the generalized He et al. (2015) initialization for forward propagation, which we used in our experiments. Derivations for the generalized Glorot and Bengio (2010); He et al. (2015) initialization, in the forward, backward and average use cases can be found in the extended version of our paper †.

We propose to initialize every weight $w_{ij}^{[\ell]} \in W^{n^{[\ell]} \times n^{[\ell-1]}}$ in a sparse layer ℓ with $n^{[\ell]}$ neurons, and connectivity mask $[m_{ij}^{[\ell]}] = M^{\ell} \in [0,1]^{n^{[\ell]} \times n^{[\ell-1]}}$ with,

$$w_{ij}^{[\ell]} \sim \mathcal{N}\left(0, \frac{2}{\textit{fan-in}_i^{[\ell]}}\right), \tag{1}$$

*We omit learning rate for simplicity and ensure different methods have same learning rate schedules when compared.

†arxiv/2010.03533



(a) Dense Layer (b) Sparse Layer (c) Signal Propagation at Init.

Figure 1: Glorot/He Initialization for a Sparse NN. All neurons in a dense NN layer (a) have the same fan-in, whereas in a sparse NN (b) the fan-in can differ for *every neuron*, potentially requiring sampling from a different distribution for every neuron. (c) Std. dev. of the pre-softmax output of LeNet5 with input sampled from a normal distribution, over 5 different randomly-initialized sparse NN for a range of sparsities.

where $\textit{fan-in}_i^{[\ell]} = \sum_{j=1}^{n^{[\ell-1]}} m_{ij}^{[\ell]}$ is the number of incoming connections for neuron i in layer ℓ.

In the special case of a dense layer where $m_{ij}^{[\ell]} = 1, \forall i, j$, Eq. (1) reduces to the initialization proposed by He et al. (2015) since $\textit{fan-in}_i^{[\ell]} = n^{[\ell-1]}, \forall i$. Similarly, the initialization proposed by Liu et al. (2019) is another special case where it is assumed $\textit{fan-in}_i^{[\ell]} \equiv \mathbb{E}_k[\textit{fan-in}_k^{[\ell]}]$, i.e. all neurons have the same number of incoming connections in a layer which is often not true with unstructured sparsity. Using the dense initialization in a sparse DNN causes signal to vanish, as empirically observed in Fig. 1c, whereas sparsity-aware initialization techniques (ours and Liu et al. (2019)) keep the variance of the signal constant.

3.2 Dynamic Sparse Training

While initialization is important for the first training step, the gradient flow during the early stages of training is not well addressed by initialization alone, rather it has been addressed in dense DNNs by normalization methods (Ioffe and Szegedy 2015). Our findings show that even with BatchNorm however, the gradient flow during the training of unstructured sparse NNs is poor.

Recently, a promising new approach to training sparse NNs has emerged — Dynamic Sparse Training (DST) — that learns connectivity adaptively during training, showing significant improvements over baseline methods that use fixed masks. These methods perform periodic updates on the sparse connectivity of each layer: commonly replacing least magnitude connections with new connections selected using various criteria. We consider two of these methods and measure their effect on gradient flow during training: Sparse Evolutionary Training (SET) (Mocanu et al. 2018), which chooses new connections randomly and Rigged Lottery (RigL) (Evci et al. 2019), which chooses the connections with high gradient magnitude. Understanding why and how these methods achieve better results can help us in improving upon them.

3.3 Lottery Ticket Hypothesis

A recent approach for training unstructured sparse NNs while achieving similar generalization to the original dense solution is the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin 2019). Notably, rather than training a pruned NN structure from random initialization, the LTH uses the dense initialization from which the pruning solution was derived.

Definition [Lottery Ticket Hypothesis]: Given a NN architecture f with parameters θ and an optimization function $O^N(f, \theta) = \theta^N$, which gives the optimized parameters of f after N training steps, there exists a sparse sub-network characterized by the binary mask M such that for some iteration K, $O^N(f, \theta^K * M)$ performs as well as $O^N(f, \theta) * M$, whereas the model trained from another random initialization θ_S, using the same mask $O^N(f, \theta_S * M)$, typically does not*. Initial results of Frankle and Carbin (2019) showed the LTH held for $K = 0$, but later results (Liu et al. 2019; Frankle et al. 2019) showed a larger K is necessary for larger datasets and NN architectures, i.e. $N \gg K \geq 0$.

We measure the gradient flow of LTs and observe poor gradient flow overall. Despite this, LTs enjoy significantly faster convergence compared to regular NN training, which motivates our investigation of the success of LTs beyond gradient flow. LTs require the connectivity mask as found by the pruning solution along with parameter values from the early part of the dense training. Given the importance of the early phase of training (Frankle, Schwab, and Morcos 2020; Lewkowycz et al. 2020), it is natural to ask about the difference between LTs and the solution they are derived from (i.e. pruning solutions). Answering this question can help us understand if the success of LTs is primarily due to its relation to the solution, or if we can identify generalizable characteristics that help with sparse NNs training.

4 Experiments

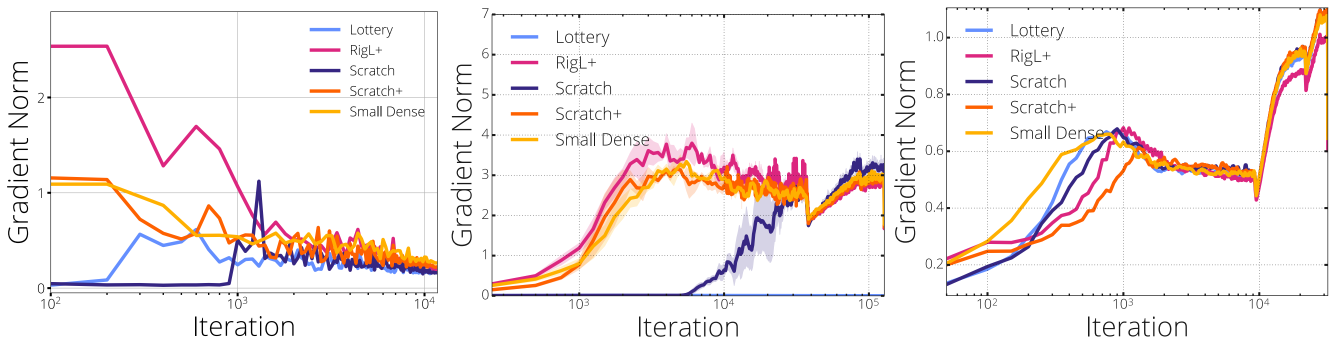
Here we show empirically that (1) sparsity-aware initialization improves gradient flow at initialization for all meth-

*See Frankle et al. (2019) for details. $*$ indicates element-wise multiplication, respecting the mask.

	MNIST			ImageNet-2012					
	LeNet5 (95% sparse)			VGG16 (80% sparse)			ResNet50 (80% sparse)		
Baseline	99.21±0.07			69.25±0.13			76.75±0.12		
Lottery	98.26±0.27			0.10±0.01			75.75±0.12*		
Small Dense	98.21±0.46			61.75±0.09			71.95±0.24		
	Dense	Sparse (Liu)	Sparse (Ours)	Dense	Sparse (Liu)	Sparse(Ours)	Dense	Sparse (Liu)	Sparse (Ours)
Scratch	62.99±42.16	96.64±0.83	**97.70**±0.09	51.81±3.02	**62.71**±0.05	62.52±0.10	70.58±0.18	**70.72**±0.16	70.63±0.22
SET	63.33±42.44	97.77±0.31	**98.16**±0.06	53.55±1.03	**63.19**±0.26	63.13±0.15	**72.93**±0.27	72.77±0.27	72.56±0.14
RigL	80.82±34.74	**98.14**±0.17	**98.13**±0.09	37.15±26.20	**63.69**±0.02	63.56±0.06	**74.41**±0.05	74.38±0.10	74.38±0.01

* With late-rewinding (i.e. $K = 5000$).

Table 1: Results of Trained Sparse/Dense Models from Different Initializations. The initialization proposed in Eq. (1) (Ours) and Liu et al. (2019) improve generalization consistently over masked dense (Original) except for in ResNet50. Note that VGG16 trained without a sparsity-aware initialization fails to converge in some instances. *Baseline* corresponds to the original dense architecture, whereas *Small Dense* corresponds to a smaller dense model with approx. the same parameter count as the sparse models.



(a) LeNet5 (b) VGG-16 (c) ResNet-50

Figure 2: Gradient Flow of Sparse Models during Training. Gradient flow during training averaged over multiple runs, '+' indicates training runs with our proposed sparse initialization and Small Dense corresponds to training of a dense network with same number of parameters as the sparse networks. Lottery ticket runs for ResNet-50 include late-rewinding.

ods, and achieves higher generalization for networks without BatchNorm, (2) the mask updates of DST methods increase gradient flow and create new negative eigenvalues in the Hessian; which we believe to be the main factor for improved generalization [‡], (3) lottery tickets have poor gradient flow, however they achieve good performance by effectively relearning the pruning solution, meaning they do not address the problem of training sparse NNs in general. Our experiments include the following settings: LeNet5 on MNIST, VGG16 on ImageNet-2012 and ResNet-50 on ImageNet-2012. Experimental details can be found in in the extended version of our work[§].

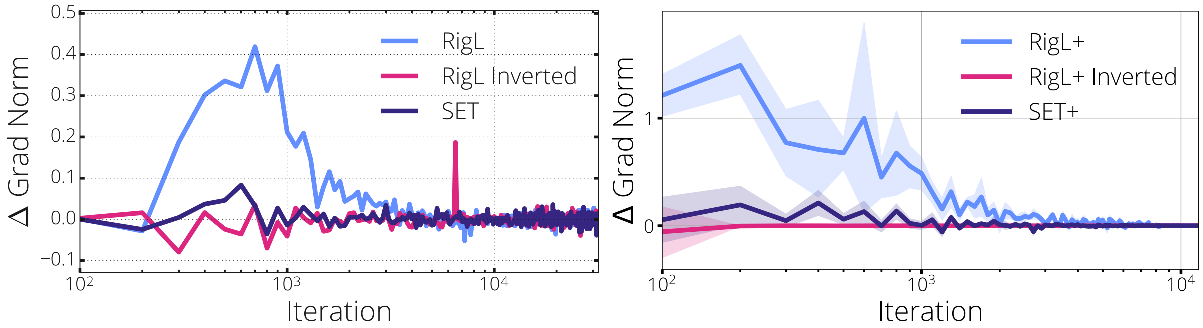
4.1 Gradient Flow at Initialization

In this section, we measure the gradient flow over the course of the training (Fig. 2) and evaluate the performance of generalized He initialization method (Table 1), and that proposed by Liu et al. (2019), over the commonly used dense initialization in sparse NN. Sparse NNs initialized using the initialization distribution of a dense model (*Scratch* in Fig. 2) start in a flat region where gradient flow is very small and thus initial progress is limited. Learning starts after 1000 iterations for LeNet5 and 5000 for VGG-16, however, generalization is sub-optimal. Liu et al. (2019) reported their proposed initialization has *no empirical effect* as compared to the masked dense initialization[¶]. In contrast, our results show their method to be largely as effective as our proposed initialization, despite the Liu et al. (2019) initialization being incorrect in the general case (see §3.1). This indicates that the assumption of a neuron having roughly uniform connectivity is sufficient for the ranges of sparsity considered, possibly due to a law-of-large-numbers-like averaging effect. We expect this effect to disappear with higher sparsity and our initialization to be more important. Results in Table 2 for a 98% sparse LeNet5 demonstrate this, where our sparsity-aware initialization shows a 40% improvement in test accuracy. Our initialization remedies the vanishing gradient problem at initialization (*Scratch+* in Fig. 2) and results in better generalization for all methods. For example, our improved initialization results in an 11% improvement in Top-1 accuracy for VGG16 (62.52 vs 51.81).

[‡]Hessian experiments are presented in the extended version

[§]Implementation of our sparse initialization and code for reproducing our experiments can be found at https://github.com/google-research/rigl/tree/master/rigl/rigl_tf2.

[¶]Models with BatchNorm and skip connections are less affected by initialization, and this is likely why the authors did not observe this effect.



(a) ResNet-50 (b) LeNet5

Figure 3: Effect of Mask Updates in Dynamic Sparse Training. Effect of mask updates on the gradient norm. *RigL Inverted* chooses connections with least magnitude. We measure the gradient norm before and after the mask updates and plot the Δ. '+' indicates proposed initialization and used in MNIST experiments.

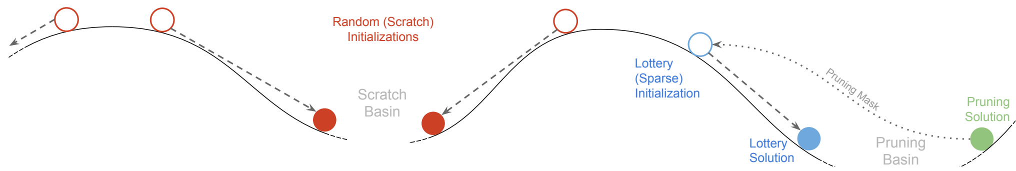


Figure 4: Lottery Tickets Are Biased Towards the Pruning Solution, Unlike Random Initialization. A cartoon illustration of the loss landscape of a sparse model, after it is pruned from a dense solution to create a LT sub-network. A LT initialization is within the basin of attraction of the pruned model's solution. In contrast a random initialization is unlikely to be close to the dense solution's basin.

	Dense/Other	Sparse (Liu)	Sparse (Ours)
Scratch	11.35±0.00	56.51±36.90	**90.00±4.04**
RigL	11.35±0.00	62.34±41.63	**96.72±0.23**
Prune	96.50±0.31	–	–

Table 2: Sparsity-aware Initialization and Generalization: Sparse (98%) LeNet5 (MNIST). Results show the importance of our sparsity-aware initialization at higher sparsity regimes, compared with a standard dense initialization (He et al. 2015), or scaled-sparse initialization (Liu et al. 2019).

While initialization is extremely important for NNs without BatchNorm and skip connections, its effect on modern architectures, such as Resnet-50, is limited (Evci et al. 2019; Zhang, Dauphin, and Ma 2019; Frankle et al. 2020b). We confirm these observations in our ResNet-50 experiments in which, despite some initial improvement in gradient flow, sparsity-aware initializations seem to have no effect on final generalization. Additionally, we observe significant increase in gradient norm after each learning rate drop (due to increased gradient variance), which suggests studying the gradient norm in the latter part of the training might not be helpful.

The LT hypothesis holds for MNIST and ResNet50 (when K=5000) but not for VGG16. We observe poor gradient flow for LTs at initialization similar to *Scratch*. After around 2000 steps gradients become non zero for *Scratch*, while gradient flow for LT experiments stay constant. Our sparse initialization improves gradient flow at initialization and we observe a significant difference in gradient flow during early training between RigL and Scratch+.

4.2 Gradient Flow During Sparse Training

Our hypothesis for Fig. 2 is that the DST methods improve gradient flow through the updates they make on the sparse connectivity, which in turn results in better performance. To verify our hypothesis, we measure the change in gradient flow whenever the sparse connectivity is updated. We also run the inverted baseline for RigL (*RigL Inverted*), in which the growing criteria is reversed and connections with the *smallest* gradients are activated as in Frankle et al. (2020b).

DST methods such as RigL replace low saliency connections during training. Assuming the pruned connections indeed have a low impact on the loss, we might expect to see increased gradient norm after new connections are activated, especially in the case of RigL, which picks new connections with high magnitude gradients. In Fig. 3 we confirm that RigL updates increase the norm of the gradient significantly, especially in the first half of training, whereas SET, which picks new connections randomly, seems to be less effective at this. Using the inverted RigL criteria doesn't improve the gradient flow, as expected, and without this RigL's performance degrades (73.83±0.12 for ResNet-50 and 92.71±7.67 for LeNet5). These results suggest that improving gradient flow early in training — as RigL does — might be the key for training sparse networks. Various recent works that improve DST methods support this hypothesis: longer update intervals Liu et al. (2021c) and adding parallel dense components

(Price and Tanner 2021) or activations (Curci, Mocanu, and Pechenizkiy 2021) that improve gradient flow brings better results.

4.3 Why Lottery Tickets are Successful

We found that LTs do not improve gradient flow, either at initialization, or early in training, as shown in Fig. 2. This may be surprising given the apparent success of LTs, however the questions posed in §3.3 present an alternative hypothesis for the ease of training from a LT initialization: Can the success of LTs be due to their relationship to well-performing pruning solutions? Here we present results showing that indeed (1) LTs initializations are consistently closer to the pruning solution than a random initialization, (2) trained LTs (i.e. LT solutions) consistently end up in the same basin as the pruning solution and (3), LT solutions are highly similar to pruning solutions under various function similarity measures. Our resulting understanding of LTs in the context of the pruning solution and the loss landscape is illustrated in Fig. 4.

Experimental Setup To investigate the relationship between the pruned and LT solutions we perform experiments on two models/datasets: a 95% sparse LeNet5 architecture (LeCun et al. 1989) with ReLU activations trained on MNIST (where the original LT formulation works, i.e. $K = 0$), and an 80% sparse ResNet-50 (Wu, Zhong, and Liu 2018) on ImageNet-2012 (Russakovsky et al. 2015) (where $K = 0$ doesn't work (Frankle et al. 2019)), for which we use values from $K = 2000$ ($\approx 6^{\text{th}}$ epoch). In both cases, we find a LT initialization by pruning each layer of a dense NN separately using magnitude-based iterative pruning (Zhu and Gupta 2018).

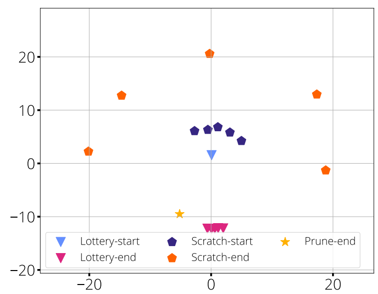
Lottery Tickets Are Close to the Pruning Solution We train 5 different models using different seeds from both *scratch* (random) and LT initializations, the results of which are in Figs. 5b and 5e. These networks share the same pruning mask and therefore lie in the same solution space. We visualize distances between initial and final points of these experiments in Figs. 5a and 5d using 2D Multi-dimensional Scaling (MDS) (Kruskal 1964) embeddings. ***LeNet5/MNIST***: In Fig. 5b, we provide the average L2 distance to the pruning solution at initialization (d_{init}), and after training (d_{final}). We observe that LT initializations start significantly closer to the pruning solution on average ($d_{init} = 13.61$ v.s. 17.46). After training, LTs end up more than $3\times$ closer to the pruning solution compared to scratch. ***Resnet-50/ImageNet-2012:*** We observe similar results for Resnet-50/ImageNet-2012. LTs, again, start closer to the pruning solution, and solutions are $5\times$ closer ($d_{final} = 39.35$ v.s. 215.98). These results explain non-random initial loss values for LT initializations (Zhou et al. 2019) and their inability to find good solutions if repelled by pruning solutions (Maene, Li, and Moens 2021). LTs are biased towards the pruning solution they are derived from, but are they in the same basin? Can it be the case that LTs learn significantly different solutions each time in function space despite being linearly connected to the pruning solution?

Lottery Tickets are in the Pruning Solution Basin Investigating paths between different solutions is a popular tool for understanding how various points in parameter space relate to each other in the loss landscape (Goodfellow, Vinyals, and Saxe 2015; Garipov et al. 2018; Draxler et al. 2018; Evci et al. 2019; Fort, Hu, and Lakshminarayanan 2020; Frankle et al. 2020a). For example, Frankle et al. (2019) use linear interpolations to show that LTs always go to the same basin[‖] when trained in different data orders. In Figs. 5c and 5f we look at the linear paths between pruning solution and 4 other points: LT initialization/solution and random (scratch) initialization/solution. Each experiment is repeated 5 times with different random seeds, and mean values are provided with 80% confidence intervals. In both experiments we observe that the linear path between LT initialization and the pruning solution decreases faster compared to the path that originates from scratch initialization. After training, the linear paths towards the pruning solution change drastically. The path from the scratch solution depicts a loss barrier; the scratch solution seems to be in a different basin than the pruning solution. In contrast, LTs are linearly connected to the pruning solution in both small and large-scale experiments indicating that LTs have the same basin of attraction as the pruning solutions they are derived from. While it seems likely, these results do not however explicitly show that the LT and pruning solutions have learned similar functions.

Lottery Tickets Learn Similar Functions to the Pruning Solution Fort, Hu, and Lakshminarayanan (2020) motivate deep ensembles by empirically showing that models starting from different random initializations typically learn different solutions, as compared to models trained from similar initializations, and thus improve performance. In Appendix A we adopt the analysis of (Fort, Hu, and Lakshminarayanan 2020), but in comparing LT initializations and random initializations using *fractional disagreement* — the fraction of class predictions over which the LT and scratch models disagree with the pruning solution they were derived from, Kullback–Leibler Divergence (KL), and Jensen–Shannon Divergence (JSD). Results in Table 3 suggest LTs models converge on a solution almost identical to the pruning solution and therefore an ensemble of LTs brings marginal improvements. Our results also show that having a fixed initialization alone can not explain the low disagreement observed for LT experiments as *Scratch* solutions obtain an average disagreement of 0.0316 despite using the same initialization, which is almost 10 times larger than that of the LT solutions (0.0043). As a result, LT ensembles show significantly less gains on MNIST and ImageNet-2012 (+0.06% and +0.54% respectively) compared to a random sparse initialization (+0.96% and +2.89%).

Implications: (a) Rewinding of LTs. Frankle et al. (2019, 2020a) argued that LTs work when the training is *stable*, and thus converges to the same basin when trained with different data sampling orders. In §4.3, we show that this basin is the same one found by pruning, and since the training converges

[‖]We define a basin as a set of points, each of which is linearly connected to at least one other point in the set.

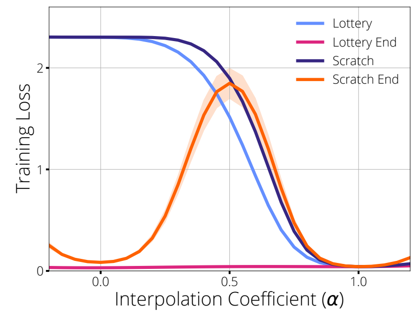


(a) MDS Projection

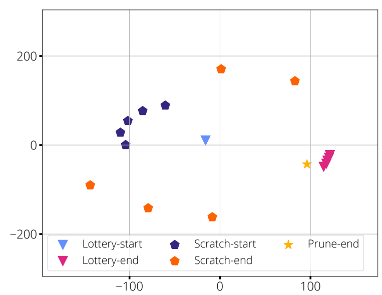
MNIST/LeNet5

Method	LT	Scratch
Acc_{test}	98.52	97.19
d_{init}	13.61	17.46
d_{final}	8.03	25.64

(b) L2 Distances



(c) Loss Interpolation

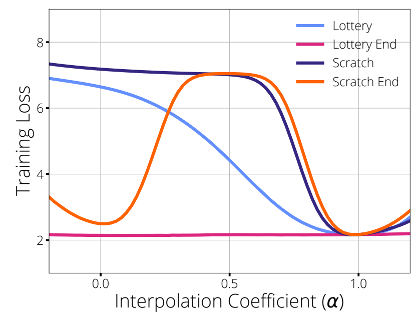


(d) MDS Projection

ImageNet-2012 /ResNet-50

Method	LT	Scratch
Acc_{test}	75.74	71.16
d_{init}	147.12	200.44
d_{final}	39.35	215.98

(e) L2 Distances



(f) Loss Interpolation

Figure 5: MDS Embeddings/L2 Distances: (a, d): 2D Multi-dimensional Scaling (MDS) embedding of sparse NNs with the same connectivity/mask; (b, e): the average L2-distance between a pruning solution and other derived sparse networks; (c, f): linear path between the pruning solution ($\alpha = 1.0$) and LT/scratch at both initialization, and solution (end of training). Top and bottom rows are for MNIST/LeNet5 and ImageNet-2012/ResNet-50 respectively.

to the same basin as before, we expect to see limited gains from rewinding if any. This is partially confirmed by Renda, Frankle, and Carbin (2020) which shows that restarting the learning rate schedule from the pruning solution performs better than rewinding the weights. **(b) Transfer of LTs.** Given the close relationship between LTs and pruning solutions, the observation that LTs trained on large datasets transfer to smaller ones, but not *vice versa* (Morcos et al. 2019; Sabatelli, Kestemont, and Geurts 2020) can be explained by a common observation in transfer learning: networks trained in large datasets transfer to smaller ones. **(c) LT's Robustness to Perturbations.** Zhou et al. (2019); Frankle, Schwab, and Morcos (2020) found that certain perturbations, like only using the signs of weights at initialization, do not impact LT generalization, while others, like shuffling the weights, do. Our results bring further insights to these observations: As long as the perturbation is small enough such that a LT stays in the same basin of attraction, results will be as good as the pruning solution. **(d) Success of LTs.** While it is exciting to see widespread applicability of LTs in different domains (Brix, Bahar, and Ney 2020; Li et al. 2020; Venkatesh et al. 2020), the results presented in this paper suggest this success may be due to the underlying pruning algorithm (and transfer learning) rather than LT initializations themselves.

5 Conclusion

In this work we studied (1) why training unstructured sparse networks from random initialization performs poorly and; (2) what makes LTs and DST the exceptions? We identified that randomly initialized unstructured sparse NNs exhibit poor gradient flow when initialized naively and proposed an alternative initialization that scales the initial variance for each neuron separately. Furthermore we showed that modern sparse NN architectures are more sensitive to poor gradient flow during early training rather than initialization alone. We observed that this is somewhat addressed by state-of-the-art DST methods, such as RigL, which significantly improves gradient flow during early training over traditional sparse training methods. Finally, we show that LTs do not improve gradient flow at either initialization or during training, but rather their success lies in effectively re-learning the original pruning solution they are derived from. We showed that a LTs initialization resides within the same basin of attraction as the pruning solution and, furthermore, when trained the LT solution learns a highly similar solution to the pruning solution, limiting their ensemble performance. These findings suggest that LTs are fundamentally limited in their potential for improving the training of sparse NNs more generally.

A Comparing Function Similarity

Fort, Hu, and Lakshminarayanan (2020) motivate deep ensembles by empirically showing that models starting from different random initializations typically learn different solutions, as compared to models trained from similar initializations. Here we adopt the analysis of (Fort, Hu, and Lakshminarayanan 2020), but in comparing LT initializations and

	Initialization	(Top-1) Test Acc.	Ensemble	Disagreement	Disagree. w/ Pruned
LeNet5 MNIST	LT	98.52 ± 0.02	98.58	0.0043 ± 0.0006	0.0089 ± 0.0002
	Scratch	97.04 ± 0.15	98.00	0.0316 ± 0.0023	0.0278 ± 0.0020
	Scratch (Diff. Init.)	97.19 ± 0.33	98.43	0.0352 ± 0.0037	0.0278 ± 0.0032
	Prune Restart	98.60 ± 0.01	98.63	0.0027 ± 0.0003	0.0077 ± 0.0003
	Pruned Soln.	98.53	–	–	–
	5 Diff. Pruned	98.30 ± 0.23	99.07	0.0214 ± 0.0023	0.0197 ± 0.0019*
ResNet50 ImageNet	LT	75.73 ± 0.08	76.27	0.0894 ± 0.0009	0.0941 ± 0.0009
	Scratch	71.16 ± 0.13	74.05	0.2039 ± 0.0013	0.2033 ± 0.0012
	Pruned Soln.	75.60	–	–	–
	5 Diff. Pruned	75.65 ± 0.13	77.80	0.1620 ± 0.0008	0.1623 ± 0.0011*

* Here we compare 4 different pruned models with the pruning solution LT/Scratch are derived from.

Table 3: Ensemble & Prediction Disagreement. We compare the function similarity (Fort, Hu, and Lakshminarayanan 2020) with the original pruning solution and ensemble generalization over 5 sparse models, trained from random initializations and LTs. As a baseline, we also show results for 5 pruned models trained from different random initializations.

random initializations using *fractional disagreement* — the fraction of class predictions over which the LT and scratch models disagree with the pruning solution they were derived from. In Table 3 we show the mean fractional disagreement over all pairs of models. We run two versions of scratch training: (1) *Scratch (Diff. Init.* different weight initialization and different data order (2) *Scratch* same weight initialization and different data order for 5 different seeds the experiments are ran. Finally, we restart training starting from the pruning solution (*Prune Restart*) using, again, 5 different data orders.

The results presented in Table 3 suggest that all 5 LTs models converge on a solution almost identical to the pruning solution. Interestingly, the 5 LT models are even more similar to each other (*Disagree.* column) than the pruning solution, possibly because they share an initialization and training is stable (Frankle et al. 2019). The disagreement of *Prune Restart* solutions with the original pruning solution matches the disagreement of lottery solutions; showing the extent of similarity between LT and pruning solutions.

Our results show that having a fixed initialization alone can not explain the low disagreement observed for LT experiments as *Scratch* solutions obtain an average disagreement of 0.0316 despite using the same initialization, which is almost 10 times more than the LT solutions (0.0043). Finding different LT initialization is costly, however using a different initialization in *Scratch (Diff. Init.)* training is free as the initializations are random. Using different initializations we can obtain more diverse solutions and thus achieve higher ensemble accuracy. As suggested by the analysis of Fort, Hu, and Lakshminarayanan (2020), ensembles of different solutions are more robust, and generalize better, than ensembles of similar solutions. An ensemble of 5 LT models with low disagreement doesn't significantly improve generalization as compared to an ensemble of 5 different pruning solutions with similar individual test accuracy. We further demonstrate these results by comparing the output probability distributions using the KL, and JSD in our extended version.

Acknowledgements

We like to thank members of the Google Brain team for their useful feedback and support. Specifically, we thank Sara Hooker, Fabian Pedregosa, Laura Graesser, Ilya Tolstikhin, Erich Elsen and Danny Tarlow for providing feedback on the preprint.

References

Brix, C.; Bahar, P.; and Ney, H. 2020. Successfully Applying the Stabilized Lottery Ticket Hypothesis to the Transformer Architecture. arXiv:2005.03454.

Chen, T.; Cheng, Y.; Gan, Z.; Yuan, L.; Zhang, L.; and Wang, Z. 2021. Chasing Sparsity in Vision Transformers: An End-to-End Exploration. arXiv:2106.04533.

Curci, S.; Mocanu, D. C.; and Pechenizkiy, M. 2021. Truly Sparse Neural Networks at Scale. arXiv:2102.01732.

Dettmers, T.; and Zettlemoyer, L. 2019. Sparse Networks from Scratch: Faster Training without Losing Performance. arXiv:1907.04840.

Dietrich, A.; Gressmann, F.; Orr, D.; Chelombiev, I.; Justus, D.; and Luschi, C. 2021. Towards Structured Dynamic Sparse Pre-Training of BERT. arXiv:2108.06277.

Draxler, F.; Veschgini, K.; Salmhofer, M.; and Hamprecht, F. A. 2018. Essentially No Barriers in Neural Network Energy Landscape. In *International Conference on Machine Learning*.

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the Lottery: Making All Tickets Winners. In *Proceedings of Machine Learning and Systems 2020*, 471–481. PMLR.

Evci, U.; Pedregosa, F.; Gomez, A. N.; and Elsen, E. 2019. The Difficulty of Training Sparse Neural Networks. arXiv:1906.10732.

Fort, S.; Hu, H.; and Lakshminarayanan, B. 2020. Deep Ensembles: A Loss Landscape Perspective. In *International Conference on Learning Representations*.

Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *7th International Conference on Learning Representations (ICLR)*.

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2019. Stabilizing the Lottery Ticket Hypothesis. arXiv:1903.01611.

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2020a. Linear Mode Connectivity and the Lottery Ticket Hypothesis. In *Proceedings of the International Conference on Machine Learning*.

Frankle, J.; Dziugaite, G. K.; Roy, D. M.; and Carbin, M. 2020b. Pruning Neural Networks at Initialization: Why are We Missing the Mark? arXiv:2009.08576.

Frankle, J.; Schwab, D. J.; and Morcos, A. S. 2020. The Early Phase of Neural Network Training. In *International Conference on Learning Representations*.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The State of Sparsity in Deep Neural Networks. arXiv:1902.09574.

Garipov, T.; Izmailov, P.; Podoprikhin, D.; Vetrov, D. P.; and Wilson, A. G. 2018. Loss Surfaces, M Connectivity, and Fast Ensembling of DNNs. In *Advances in Neural Information Processing Systems*.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Golubeva, A.; Neyshabur, B.; and Gur-Ari, G. 2021. Are wider nets better given the same number of parameters? arXiv:2010.14495.

Goodfellow, I. J.; Vinyals, O.; and Saxe, A. M. 2015. Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *IEEE International Conference on Computer Vision (ICCV)*.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*.

Kruskal, J. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*.

Kusupati, A.; Ramanujan, V.; Somani, R.; Wortsman, M.; Jain, P.; Kakade, S.; and Farhadi, A. 2020. Soft Threshold Weight Reparameterization for Learnable Sparsity. In *Proceedings of the International Conference on Machine Learning*.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computation*.

Lee, N.; Ajanthan, T.; Gould, S.; and Torr, P. H. S. 2020. A Signal Propagation Perspective for Pruning Neural Networks at Initialization. In *ICLR*.

Lee, N.; Ajanthan, T.; and Torr, P. H. S. 2019. SNIP: Single-shot Network Pruning based on Connection Sensitivity. In *ICLR*.

Lewkowycz, A.; Bahri, Y.; Dyer, E.; Sohl-Dickstein, J.; and Gur-Ari, G. 2020. The large learning rate phase of deep learning: the catapult mechanism. arXiv:2003.02218.

Li, B.; Wang, S.; Jia, Y.; Lu, Y.; Zhong, Z.; Carin, L.; and Jana, S. 2020. Towards Practical Lottery Ticket Hypothesis for Adversarial Training. arXiv:2003.05733.

Liu, S.; Chen, T.; Atashgahi, Z.; Chen, X.; Sokar, G.; Mocanu, E.; Pechenizkiy, M.; Wang, Z.; and Mocanu, D. C. 2021a. FreeTickets: Accurate, Robust and Efficient Deep Ensemble by Training with Dynamic Sparsity. arXiv:2106.14568.

Liu, S.; Ni'mah, I.; Menkovski, V.; Mocanu, D.; and Pechenizkiy, M. 2021b. Efficient and effective training of sparse recurrent neural networks. *Neural Comput. Appl.*, 33: 9625–9636.

Liu, S.; Yin, L.; Mocanu, D.; and Pechenizkiy, M. 2021c. Do We Actually Need Dense Over-Parameterization? In-Time Over-Parameterization in Sparse Training. In *ICML*.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2019. Rethinking the Value of Network Pruning. In *International Conference on Learning Representations*.

Louizos, C.; Ullrich, K.; and Welling, M. 2017. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*.

Lubana, E. S.; and Dick, R. 2021. A Gradient Flow Framework For Analyzing Network Pruning. In *International Conference on Learning Representations (ICLR), 2021*.

Maene, J.; Li, M.; and Moens, M. 2021. Towards Understanding Iterative Magnitude Pruning: Why Lottery Tickets Win. arXiv:2106.06955.

Mocanu, D. C.; Mocanu, E.; Stone, P.; Nguyen, P. H.; Gibescu, M.; and Liotta, A. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*.

Molchanov, D.; Ashukha, A.; and Vetrov, D. 2017. Variational Dropout Sparsifies Deep Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR.

Morcos, A.; Yu, H.; Paganini, M.; and Tian, Y. 2019. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems 32*.

Mostafa, H.; and Wang, X. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*.

Mozer, M. C.; and Smolensky, P. 1989. Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. In *Advances in Neural Information Processing Systems 1*. Morgan-Kaufmann.

Price, I.; and Tanner, J. 2021. Dense for the Price of Sparse: Improved Performance of Sparsely Initialized Networks via a Subspace Offset. arXiv:2102.07655.

Ramanujan, V.; Wortsman, M.; Kembhavi, A.; Farhadi, A.; and Rastegari, M. 2019. What's Hidden in a Randomly Weighted Neural Network? arXiv:1911.13299.

Renda, A.; Frankle, J.; and Carbin, M. 2020. Comparing Rewinding and Fine-tuning in Neural Network Pruning. In *International Conference on Learning Representations*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*.

Sabatelli, M.; Kestemont, M.; and Geurts, P. 2020. On the Transferability of Winning Tickets in Non-Natural Image Datasets. arXiv:2005.05232.

Sokar, G.; Mocanu, E.; Mocanu, D. C.; Pechenizkiy, M.; and Stone, P. 2021. Dynamic Sparse Training for Deep Reinforcement Learning. arXiv:2106.04217.

Tanaka, H.; Kunin, D.; Yamins, D. L. K.; and Ganguli, S. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems*.

Tessera, K.; Hooker, S.; and Rosman, B. 2021. Keep the Gradients Flowing: Using Gradient Flow to Study Sparse Network Optimization. arXiv:2102.01670.

Venkatesh, B.; Thiagarajan, J. J.; Thopalli, K.; and Sattigeri, P. 2020. Calibrate and Prune: Improving Reliability of Lottery Tickets Through Prediction Calibration. arXiv:2002.03875.

Wang, C.; Zhang, G.; and Grosse, R. 2020. Picking Winning Tickets Before Training by Preserving Gradient Flow. In *ICLR*.

Wu, S.; Zhong, S.; and Liu, Y. 2018. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*.

Xiao, L.; Bahri, Y.; Sohl-Dickstein, J.; Schoenholz, S. S.; and Pennington, J. 2018. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning (ICML)*.

Zhang, H.; Dauphin, Y. N.; and Ma, T. 2019. Fixup Initialization: Residual Learning Without Normalization. In *International Conference on Learning Representations*.

Zhou, H.; Lan, J.; Liu, R.; and Yosinski, J. 2019. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*.

Zhu, M.; and Gupta, S. 2018. To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. In *International Conference on Learning Representations Workshop*.