Linear Mode Connectivity and the Lottery Ticket Hypothesis

Jonathan Frankle [1]   Gintare Karolina Dziugaite [2]   Daniel M. Roy [3 4]   Michael Carbin [1]

Abstract

We study whether a neural network optimizes to the same, linearly connected minimum under different samples of SGD noise (e.g., random data order and augmentation). We find that standard vision models become *stable to SGD noise* in this way early in training. From then on, the outcome of optimization is determined to a linearly connected region. We use this technique to study *iterative magnitude pruning* (IMP), the procedure used by work on the lottery ticket hypothesis to identify subnetworks that could have trained in isolation to full accuracy. We find that these subnetworks only reach full accuracy when they are stable to SGD noise, which either occurs at initialization for small-scale settings (MNIST) or early in training for large-scale settings (ResNet-50 and Inception-v3 on ImageNet).

1. Introduction

When training a neural network with mini-batch stochastic gradient descent (SGD), training examples are presented to the network in a random order within each epoch. In many cases, each example also undergoes random data augmentation. This randomness can be seen as *noise* that varies from training run to training run and alters the network's trajectory through the optimization landscape, even when the initialization and hyperparameters are fixed. In this paper, we investigate how this SGD noise affects the outcome of optimizing neural networks and the role this effect plays in sparse, *lottery ticket* networks (Frankle & Carbin, 2019).

Instability analysis. To study these questions, we propose *instability analysis*. The goal of instability analysis is to determine whether the outcome of optimizing a particular neural network is *stable to SGD noise*. Figure 1 (left) visualizes instability analysis. First, we create a network $\mathcal{N}$ with random initialization W_0. We then train two copies of $\mathcal{N}$
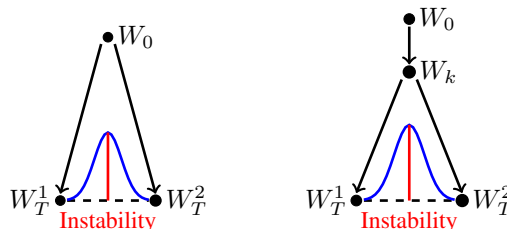
Figure 1. A diagram of instability analysis from step 0 (left) and step k (right) when comparing networks using linear interpolation.

with different samples of SGD noise (i.e., different random data orders and augmentations). Finally, we compare the resulting networks to measure the effect of these different samples of SGD noise on the outcome of optimization. If the networks are sufficiently similar according to a criterion, we determine $\mathcal{N}$ to be stable to SGD noise. We also study this behavior starting from the state of $\mathcal{N}$ at step k of training (Figure 1 right). Doing so allows us to determine *when* the outcome of optimization becomes stable to SGD noise.

There are many possible ways in which to compare the networks that result from instability analysis (Appendix G). We use the behavior of the optimization landscape along the line between these networks (blue in Figure 1). Does error remain flat or even decrease (meaning the networks are in the same, linearly connected minimum), or is there a barrier of increased error? We define the *linear interpolation instability* of $\mathcal{N}$ to SGD noise as the maximum increase in error along this path (red). We consider $\mathcal{N}$ stable to SGD noise if error does not increase along this path, i.e., instability ≈ 0. This means $\mathcal{N}$ will find the same, linearly connected minimum regardless of the sample of SGD noise.

By linearly interpolating at the end of training in this fashion, we assess a linear form of *mode connectivity*, a phenomenon where the minima found by two networks are connected by a path of nonincreasing error. Freeman & Bruna (2017), Draxler et al. (2018), and Garipov et al. (2018) show that the modes of standard vision networks trained from different initializations are connected by nonlinear paths of constant error or loss. Based on this work, we expect that all networks we examine are connected by such paths. However, the modes found by Draxler et al. and Garipov et al. are not connected by *linear* paths. The only extant example of linear mode connectivity is by Nagarajan & Kolter (2019), who train MLPs from the same initialization on disjoint

subsets of MNIST and find that the resulting networks are connected by linear paths of constant test error. In contrast, we explore linear mode connectivity from points throughout training, we do so at larger scales, and we focus on different samples of SGD noise rather than disjoint samples of data.

We perform instability analysis on standard networks for MNIST, CIFAR-10, and ImageNet. All but the smallest MNIST network are unstable to SGD noise at initialization according to linear interpolation. However, by a point early in training (3% for ResNet-20 on CIFAR-10 and 20% for ResNet-50 on ImageNet), all networks become stable to SGD noise. From this point on, the outcome of optimization is determined to a linearly connected minimum.

The lottery ticket hypothesis. Finally, we show that instability analysis and linear interpolation are valuable scientific tools for understanding other phenomena in deep learning. Specifically, we study the sparse networks discussed by the recent *lottery ticket hypothesis* (LTH; Frankle & Carbin, 2019). The LTH conjectures that, at initialization, neural networks contain sparse subnetworks that can train in isolation to full accuracy.

Empirical evidence for the LTH consists of experiments using a procedure called *iterative magnitude pruning* (IMP). On small networks for MNIST and CIFAR-10, IMP retroactively finds subnetworks at initialization that can train to the same accuracy as the full network (we call such subnetworks *matching*). Importantly, IMP finds matching subnetworks at *nontrivial* sparsity levels, i.e., those beyond which subnetworks found by trivial random pruning are matching. In more challenging settings, however, there is no empirical evidence for the LTH: IMP subnetworks of VGGs and ResNets on CIFAR-10 and ImageNet are not matching at nontrivial sparsities (Liu et al., 2019; Gale et al., 2019).

We show that instability analysis distinguishes known cases where IMP succeeds and fails to find matching subnetworks at nontrivial sparsities, providing the first basis for understanding the mixed results in the literature. Namely, IMP subnetworks are only matching when they are stable to SGD noise according to linear interpolation. Using this insight, we identify new scenarios where we can find sparse, matching subnetworks at nontrivial sparsities, including in more challenging settings (e.g., ResNet-50 on ImageNet). In these settings, sparse IMP subnetworks become stable to SGD noise *early* in training rather than at initialization, just as we find with the unpruned networks. Moreover, these stable IMP subnetworks are also matching. In other words, early in training (if not at initialization), sparse subnetworks emerge that can complete training in isolation and reach full accuracy. These findings shed new light on neural network training dynamics, hint at possible mechanisms underlying lottery ticket phenomena, and extend the lottery ticket observations to larger scales.

Contributions. We make the following contributions:

- We introduce *instability analysis* to determine whether the outcome of optimizing a neural network is stable to SGD noise, and we suggest linear mode connectivity for making this determination.

- On a range of image classification benchmarks including standard networks on ImageNet, we observe that networks become stable to SGD noise early in training.

- We use instability analysis to distinguish successes and failures of IMP (the method behind extant lottery ticket results). Namely, sparse IMP subnetworks are matching only when they are stable to SGD noise.

- We generalize IMP to find subnetworks early in training rather than at initialization. We show that IMP subnetworks become stable and matching when set to their weights from early in training, making it possible to extend the lottery ticket observations to larger scales.

2. Preliminaries and Methodology

Instability analysis. To perform instability analysis on a network $\mathcal{N}$ with weights W, we make two copies of $\mathcal{N}$ and train them with different random samples of SGD noise (i.e., different data orders and augmentations), producing trained weights W_T^1 and W_T^2. We compare these weights with a function, producing a value we call the *instability* of $\mathcal{N}$ to SGD noise. We then determine whether this value satisfies a criterion indicating that $\mathcal{N}$ is stable to SGD noise. The weights of $\mathcal{N}$ could be randomly initialized ($W = W_0$ in Figure 1) or the result of k training steps ($W = W_k$).

Formally, we model SGD by function $\mathcal{A}^{s \to t} : \mathbb{R}^D \times U \to \mathbb{R}^D$ that maps weights $W_s \in \mathbb{R}^D$ at step s and SGD randomness $u \sim U$ to weights $W_t \in \mathbb{R}^D$ at step t by training for $t - s$ steps ($s, t \in \{0, .., T\}$). Algorithm 1 outlines instability analysis with a function $f : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$.

Algorithm 1 Compute instability of W_k with function f.

1: Train W_k to W_T^1 with noise $u_1 \sim U$: $W_T^1 = \mathcal{A}^{k \to T}(W_k, u_1)$
2: Train W_k to W_T^2 with noise $u_2 \sim U$: $W_T^2 = \mathcal{A}^{k \to T}(W_k, u_2)$
3: Return $f(W_T^1, W_T^2)$, i.e., the *instability* of W_k to SGD noise.

Linear interpolation. Consider a path p on the optimization landscape between networks W_1 and W_2. We define the *error barrier height* of p as the maximum increase in error from that of W_1 and W_2 along path p. For instability analysis, we use as our function f the error barrier height along the *linear* path between two networks W_1 and W_2.[1]

Formally, let $\mathcal{E}(W)$ be the (train or test) error of a network with weights W. Let $\mathcal{E}_\alpha(W_1, W_2) = \mathcal{E}(\alpha W_1 + (1 - \alpha)W_2)$ for $\alpha \in [0, 1]$ be the error of the network created by linearly interpolating between W_1 and W_2. Let $\mathcal{E}_{\mathsf{sup}}(W_1, W_2) = \sup_\alpha \mathcal{E}_\alpha(W_1, W_2)$ be the highest error when interpolating in

---
[1]See Appendix G for alternate ways of comparing the networks.

Network	Variant	Dataset	Params	Train Steps	Batch	Accuracy	Optimizer	Rate	Schedule	Warmup	BatchNorm	Pruned Density	Style
LeNet		MNIST	266K	24K Iters	60	$98.3 \pm 0.1\%$	adam	12e-4	constant	0	No	3.5%	Iterative
ResNet-20	Standard					$91.7 \pm 0.1\%$		0.1		0		16.8%	
ResNet-20	Low	CIFAR-10	274K	63K Iters	128	$88.8 \pm 0.1\%$	momentum	0.01	10x drop at 32K, 48K	0	Yes	8.6%	Iterative
ResNet-20	Warmup					$89.7 \pm 0.3\%$		0.03		30K		8.6%	
VGG-16	Standard					$93.7 \pm 0.1\%$		0.1		0		1.5%	
VGG-16	Low	CIFAR-10	14.7M	63K Iters	128	$91.7 \pm 0.1\%$	momentum	0.01	10x drop at 32K, 48K	0	Yes	5.5%	Iterative
VGG-16	Warmup					$93.4 \pm 0.1\%$		0.1		30K		1.5%	
ResNet-50		ImageNet	25.5M	90 Eps	1024	$76.1 \pm 0.1\%$	momentum	0.4	10x drop at 30,60,80	5 Eps	Yes	30%	One-Shot
Inception-v3		ImageNet	27.1M	171 Eps	1024	$78.1 \pm 0.1\%$	momentum	0.03	linear decay to 0.005	0	Yes	30%	One-Shot

Table 1. Our networks and hyperparameters. Accuracies are the means and standard deviations across three initializations. Hyperparameters for ResNet-20 standard are from He et al. (2016). Hyperparameters for VGG-16 standard are from Liu et al. (2019). Hyperparameters for *low*, *warmup*, and LeNet are adapted from Frankle & Carbin (2019). Hyperparameters for ImageNet networks are from Google's reference TPU code (Google, 2018). Note: Frankle & Carbin mistakenly refer to ResNet-20 as "ResNet-18," which is a separate network.
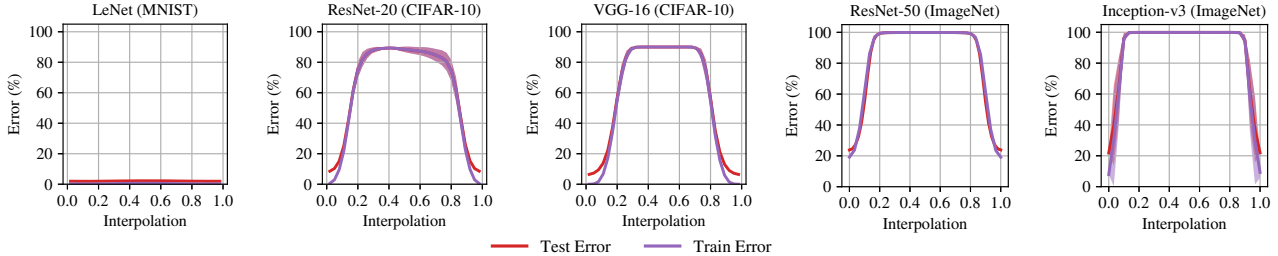

Figure 2. Error when linearly interpolating between networks trained from the same initialization with different SGD noise. Lines are means and standard deviations over three initializations and three data orders (nine samples total). Trained networks are at 0.0 and 1.0.

this way. Finally, let $\bar{\mathcal{E}}(W_1, W_2) = \mathsf{mean}(\mathcal{E}(W_1), \mathcal{E}(W_2))$. The error barrier height on the linear path between W_1 and W_2 (which is our function f for instability analysis) is $\mathcal{E}_{\mathsf{sup}}(W_1, W_2) - \bar{\mathcal{E}}(W_1, W_2)$ (red line in Figure 1). When we perform instability analysis on a network $\mathcal{N}$ with this function, we call this quantity the *linear interpolation instability* (shorthand: *instability*) of $\mathcal{N}$ to SGD noise.

Linear mode connectivity. Two networks W_1 and W_2 are *mode connected* if there exists a path between them along which the error barrier height ≈ 0 (Draxler et al., 2018; Garipov et al., 2018). They are *linearly mode connected* if this is true along the linear path. For instability analysis, we consider a network $\mathcal{N}$ stable to SGD noise (shorthand: *stable*) when the networks that result from instability analysis are linearly mode connected; that is, when the linear interpolation instability of $\mathcal{N} \approx 0$. Otherwise, it is unstable to SGD noise (shorthand: *unstable*). Empirically, we consider instability $< 2\%$ to be stable; this margin accounts for noise and matches increases in error along paths found by Draxler et al. (2018, Table B.1) and Garipov et al. (2018, Table 2). We use 30 evenly-spaced values of α, and we average instability from three initializations and three runs per initialization (nine combinations total).

Networks and datasets. We study image classification networks on MNIST, CIFAR-10, and ImageNet as listed in Table 1. All hyperparameters are standard values from reference code or prior work as cited in Table 1. The *warmup* and *low* variants of ResNet-20 and VGG-16 are adapted from hyperparameters in Frankle & Carbin (2019).

3. Instability Analysis of Unpruned Networks

In this section, we perform instability analysis on the standard networks in Table 1 from many points during training. We find that, although only LeNet is stable to SGD noise at initialization, every network becomes stable early in training, meaning the outcome of optimization from that point forward is determined to a linearly connected minimum.

Instability analysis at initialization. We first perform instability analysis from initialization. We use Algorithm 1 with W_0 (visualized in Figure 1 left): train two copies of the same, randomly initialized network with different samples of SGD noise. Figure 2 shows the train (purple) and test (red) error when linearly interpolating between the minima found by these copies. Except for LeNet (MNIST), none of the networks are stable at initialization. In fact, train and test error rise to the point of random guessing when linearly interpolating. LeNet's error rises slightly, but by less than a percentage point. We conclude that, in general, larger-scale image classification networks are unstable at initialization according to linear interpolation.

Instability analysis during training. Although larger networks are unstable at initialization, they may become stable at some point afterwards; for example, in the limit, they will be stable trivially after the last step of training. To investigate when networks become stable, we perform instability analysis using the state of the network at various training steps. That is, we train a network for k steps, make two copies, train them to completion with different samples of
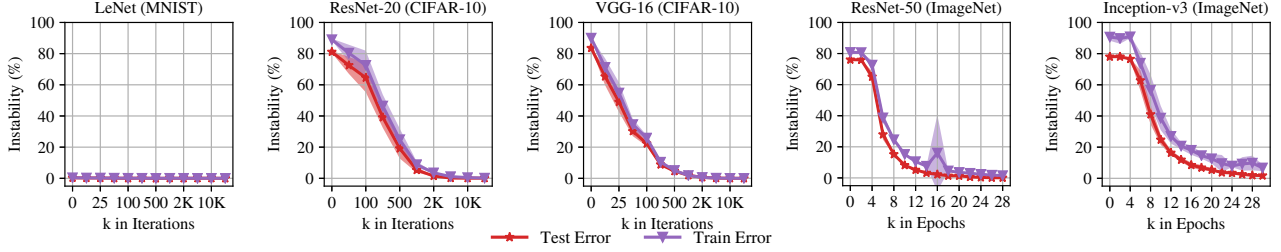
Figure 3. Linear interpolation instability when starting from step k. Each line is the mean and standard deviation across three initializations and three data orders (nine samples in total).
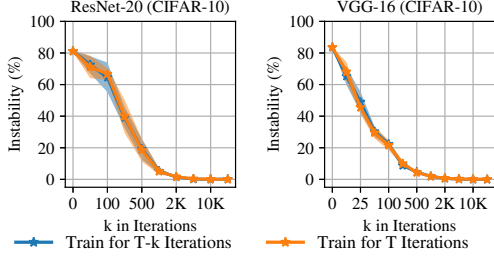

Figure 4. Linear interpolation instability on the test set when making two copies of the state of the network at step k and either (1) training for the remaining $T - k$ steps (blue) or (2) training for T steps with the learning rate schedule reset to step 0 (orange).

SGD noise, and linearly interpolate (Figure 1 right). We do so for many values of k, assessing whether there is a point after which the outcome of optimization is determined to a linearly connected minimum regardless of SGD noise.

For each k, Figure 3 shows the linear interpolation instability of the network at step k, i.e., the maximum error during interpolation (the peaks in Figure 2) minus the mean of the errors of the two networks (the endpoints in Figure 2). In all cases, test set instability decreases as k increases, culminating in stable networks. The steps at which networks become stable are early in training. For example, they do so at iterations 2000 for ResNet-20 and 1000 VGG-16; in other words, after 3% and 1.5% of training, SGD noise does not affect the final linearly connected minimum. ResNet-50 and Inception-v3 become stable later: at epoch 18 (20% into training) and 28 (16%), respectively, using the test set.

For LeNet, ResNet-20, and VGG-16, instability is essentially identical when measured in terms of train or test error, and the networks become stable to SGD noise at the same time for both quantities. For ResNet-50 and Inception-v3, train instability follows the same trend as test instability but is slightly higher at all points, meaning train set stability occurs later for ResNet-50 and does not occur in our range of analysis for Inception-v3. Going forward, we present all results with respect to test error for simplicity and include corresponding train error data in the appendices. 3

Disentangling instability from training time. Varying the step k from which we run instability analysis has two effects. First, it changes the state of the network from which we train

Algorithm 2 IMP rewinding to step k and N iterations.

1: Create a network with randomly initialization $W_0 \in \mathbb{R}^d$.
2: Initialize pruning mask to $m = 1^d$.
3: Train W_0 to W_k with noise $u \sim U$: $W_k = \mathcal{A}^{0 \to k}(W_0, u)$.
4: **for** $n \in \{1, \ldots, N\}$ **do**
5:     Train $m \odot W_k$ to $m \odot W_T$ with noise $u' \sim U$:
    $W_T = \mathcal{A}^{k \to T}(m \odot W_k, u')$.
6:     Prune the lowest magnitude entries of W_T that remain. Let $m[i] = 0$ if $W_T[i]$ is pruned.
7: Return W_k, m

Network	Full	IMP	Rand Prune	Rand Reinit	Δ	IMP Matching?
LeNet	98.3	98.2	96.7	97.5	0.1	Y
ResNet-20	91.7	88.5	88.6	88.8	3.2	N
ResNet-20 Low	88.8	89.0	85.7	84.7	-0.2	Y
ResNet-20 Warmup	89.7	89.6	85.7	85.6	0.1	Y
VGG-16	93.7	90.9	89.4	91.0	2.8	N
VGG-16 Low	91.7	91.6	90.1	90.2	0.1	Y
VGG-16 Warmup	93.4	93.2	90.1	90.7	0.2	Y
ResNet-50	76.1	73.7	73.1	73.4	2.4	N
Inception-v3	78.1	75.7	75.2	75.5	2.4	N

Table 2. Accuracy of IMP and random subnetworks when rewinding to $k = 0$ at the sparsities in Table 1. Accuracies are means across three initializations. All standard deviations are < 0.2.

two copies to completion on different SGD noise. Second, it changes the number of steps for which those copies are trained: when we run instability analysis from step k, we train the copies under different SGD noise for $T - k$ steps. As k increases, the copies have fewer steps during which to potentially find linearly unconnected minima. It is possible that the gradual decrease in instability as k increases and the eventual emergence of linear mode connectivity is just an artifact of these shorter training times.

To disentangle the role of training time in our experiments, we modify instability analysis to train the copies for T iterations no matter the value of k. When doing so, we reset the learning rate schedule to iteration 0 after making the copies. In Figure 4, we compare instability with and without this modification for ResNet-20 and VGG-16 on CIFAR-10. Instability is indistinguishable in both cases, indicating that the different numbers of training steps did not play a role in the earlier results. Going forward, we present all results by training copies for $T - k$ steps as in Algorithm 1.

4. Instability Analysis of Lottery Tickets

In this section, we leverage instability analysis and our observations about linear mode connectivity to gain new insights into the behavior of sparse *lottery ticket* networks.

4.1. Overview

We have long known that it is possible to *prune* neural networks after training, often removing 90% of weights without reducing accuracy after some additional training (e.g., Reed, 1993; Han et al., 2015; Gale et al., 2019). However, sparse networks are more difficult to train from scratch. Beyond *trivial* sparsities where many weights remain and random subnetworks can train to full accuracy, sparse networks trained in isolation are generally less accurate than the corresponding dense networks (Han et al., 2015; Li et al., 2017; Liu et al., 2019; Frankle & Carbin, 2019).

However, there is a known class of sparse networks that remain accurate at nontrivial sparsities. On small vision tasks, an algorithm called *iterative magnitude pruning* (IMP) retroactively finds sparse subnetworks that were capable of training in isolation from initialization to full accuracy at the sparsities attained by pruning (Frankle & Carbin, 2019). The existence of such subnetworks raises the prospect of replacing conventional, dense networks with sparse ones, creating new opportunities to reduce the cost of training. However, in more challenging settings, IMP subnetworks perform no better than subnetworks chosen randomly, meaning they only train to full accuracy at trivial sparsities (Liu et al., 2019; Gale et al., 2019).

We find that instability analysis offers new insights into the behavior of IMP subnetworks and a potential explanation for their successes and failures. Namely, the sparsest IMP subnetworks only train to full accuracy when they are stable to SGD noise. In other words, when different samples of SGD noise cause an IMP subnetwork to find minima that are not linearly connected, then test accuracy is lower.

4.2. Methodology

Iterative magnitude pruning. Iterative magnitude pruning (IMP) is a procedure to retroactively find a subnetwork of the state of the full network at step k of training. To do so, IMP trains a network to completion, prunes weights with the lowest magnitudes globally, and *rewinds* the remaining weights back to their values at iteration k (Algorithm 2). The result is a subnetwork (W_k, m) where $W_k \in \mathbb{R}^d$ is the state of the full network at step k and $m \in \{0, 1\}^d$ is a mask such that $m \odot W_k$ (where $\odot$ is the element-wise product) is a pruned network. We can run IMP iteratively (pruning 20% of weights (Frankle & Carbin, 2019), rewinding, and repeating until a target sparsity) or in one shot (pruning to a target sparsity at once). We one-shot prune ImageNet networks for efficiency and iteratively prune otherwise (Table 1).

Frankle & Carbin (2019) focus on finding sparse subnetworks at initialization; as such, they only use IMP to "reset" unpruned weights to their values at initialization. One of our contributions is to generalize IMP to *rewind* to any step k. Frankle & Carbin refer to subnetworks that match the accuracy of the full network as *winning tickets* because they have "won the initialization lottery" with weights that make attaining this accuracy possible. When we rewind to iteration $k > 0$, subnetworks are no longer randomly initialized, so the term *winning ticket* is no longer appropriate. Instead, we refer to such subnetworks simply as *matching*.

Sparsity levels. In this section, we focus on the most extreme sparsity levels for which IMP returns a matching subnetwork at any rewinding step k. These levels are in Table 1, and Appendix A explains these choices. These sparsities provide the best contrast between sparse networks that are matching and (1) the full, overparameterized networks and (2) other classes of sparse networks. Appendix D includes the analyses from this section for all sparsities for ResNet-20 and VGG-16, which we summarize in Section 4.4; due to the computational costs of these experiments, we only collected data across all sparsities for these networks.

4.3. Experiments and Results

Recapping the lottery ticket hypothesis. We begin by studying sparse subnetworks rewound to initialization ($k = 0$). This is the lottery ticket experiment from Frankle & Carbin (2019). As Table 2 shows, when rewinding to step 0, IMP subnetworks of LeNet are matching, as are variants of ResNet-20 and VGG-16 with lower learning rates or learning rate warmup (changes proposed by Frankle & Carbin to make it possible for IMP to find matching subnetworks). However, IMP subnetworks of standard ResNet-20, standard VGG-16, ResNet-50, and Inception-v3 are not matching. In fact, they are no more accurate than subnetworks generated by randomly pruning or reinitializing the IMP subnetworks, suggesting that neither the structure nor the initialization uncovered by IMP provides a performance advantage. For full details on the accuracy of these subnetworks at all levels of sparsity, see Appendix A.

Instability analysis of subnetworks at initialization. When we perform instability analysis on these subnetworks, we find that they are only matching when they are stable to SGD noise (Figure 5). The IMP subnetworks of LeNet, ResNet-20 (low, warmup), and VGG-16 (low, warmup) are stable and matching (Figure 5, left). In all other cases, IMP subnetworks are neither stable nor matching (Figure 5, left). The low and warmup results are notable because Frankle & Carbin selected these hyperparameters specifically for IMP to find matching subnetworks; that this change also makes the subnetworks stable adds further evidence of a connection between instability and accuracy in IMP subnetworks.
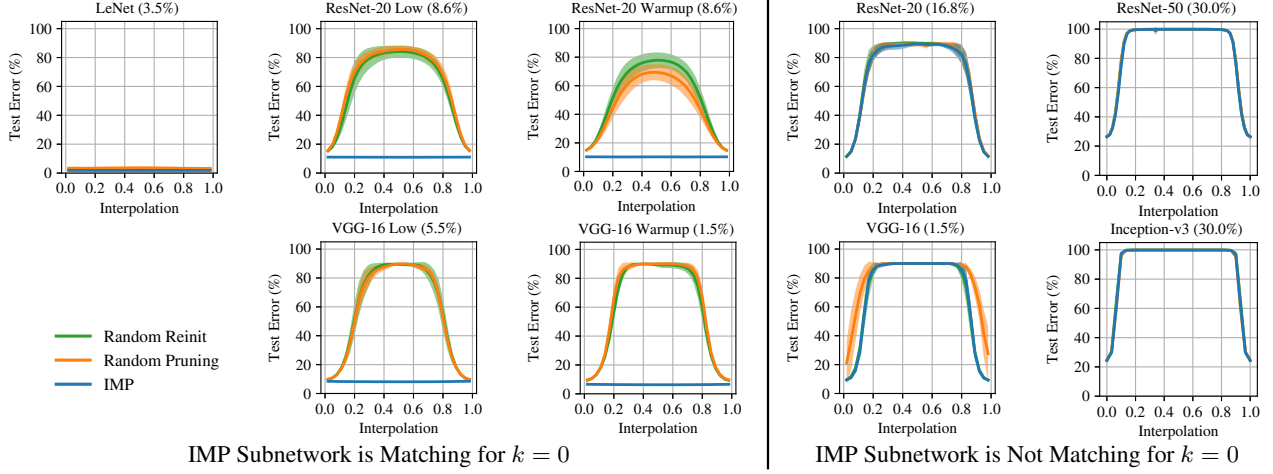
Figure 5. Test error when linearly interpolating between subnetworks trained from the same initialization with different SGD noise. Lines are means and standard deviations over three initializations and three data orders (nine in total). Percents are weights remaining.
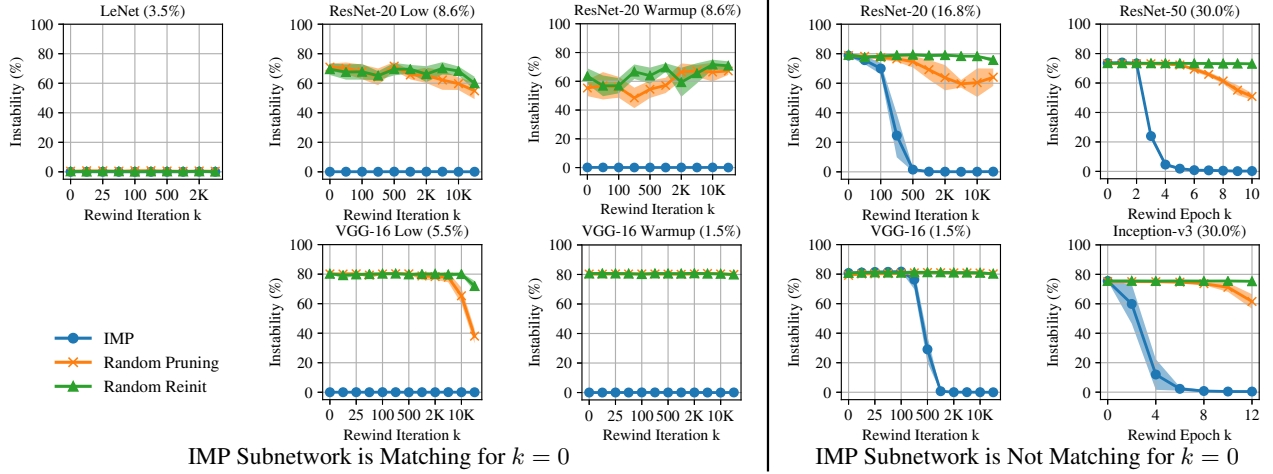

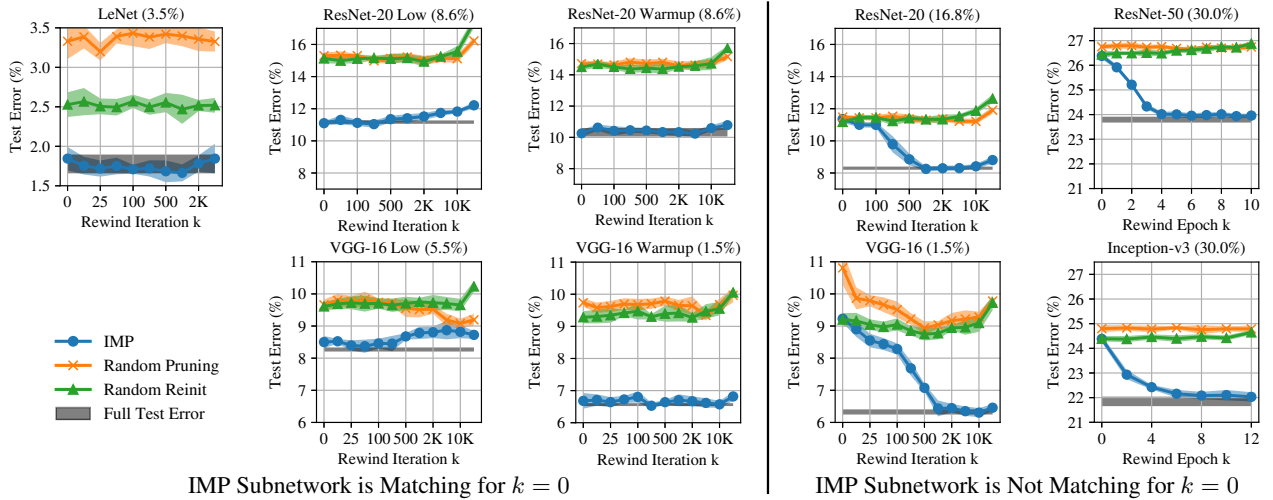*Figure 6.* Linear interpolation instability of subnetworks created using the state of the full network at step k and applying a pruning mask. Lines are means and standard deviations over three initializations and three data orders (nine in total). Percents are weights remaining.
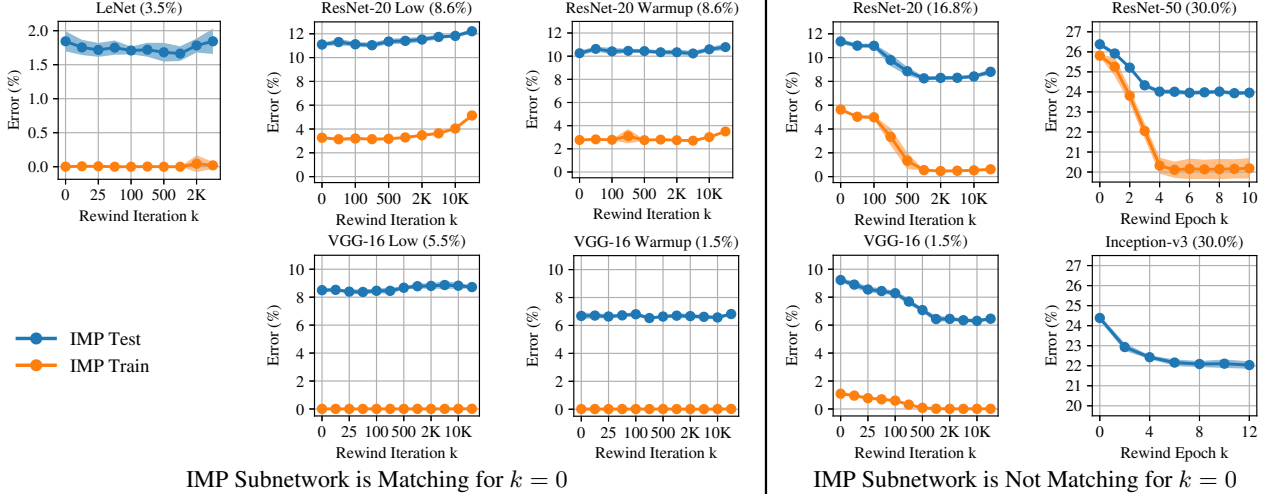

Figure 7. Test error of subnetworks created using the state of the full network at step k and applying a pruning mask. Lines are means and standard deviations over three initializations and three data orders (nine in total). Percents are weights remaining.

Figure 8. Train error of subnetworks created using the state of the full network at step k and apply a pruning mask. Lines are means and standard deviations over three initializations and three data orders (nine in total). Percents are weights remaining. We did not compute the train set quantities for Inception-v3 due to computational limitations.

No randomly pruned or reinitialized subnetworks are stable or matching at these sparsities except those of LeNet: LeNet subnetworks are not matching but error only rises slightly when interpolating. For all other networks, error approaches that of random guessing when interpolating.

Instability analysis of subnetworks during training. We just saw that IMP subnetworks are matching from initialization only when they are stable. In Section 3, we found that unpruned networks become stable only after a certain amount of training. Here, we combine these observations: we study whether IMP subnetworks become stable later in training and, if so, whether improved accuracy follows.

Concretely, we perform IMP where we rewind to iteration $k > 0$ after pruning. Doing so produces a subnetwork (W_k, m) of the state of the full network at iteration k. We then run instability analysis using this subnetwork. Another way of looking at this experiment is that it simulates training the full network to iteration k, generating a pruning mask, and evaluating the instability of the resulting subnetwork; the underlying mask-generation procedure involves training the network many times in the course of performing IMP.

The blue dots in Figure 6 show the instability of the IMP subnetworks at many rewinding iterations. Networks whose IMP subnetworks were stable when rewinding to iteration 0 remain stable at all other rewinding points (Figure 6, left). Notably, networks whose IMP subnetworks were *unstable* when rewinding to iteration 0 become stable when rewinding later. IMP subnetworks of ResNet-20 and VGG-16 become stable at iterations 500 (0.8% into training) and 1000 (1.6%). Likewise, IMP subnetworks of ResNet-50 and Inception-v3 become stable at epochs 5 (5.5% into training) and 6 (3.5%). In all cases, the IMP subnetworks become stable sooner

than the unpruned networks, substantially so for ResNet-50 (epoch 5 vs. 18) and Inception-v3 (epoch 6 vs. 28).

The test error of the IMP subnetworks behaves similarly. The blue line in Figure 7 plots the error of the IMP subnetworks and the gray line plots the error of the full networks to one standard deviation; subnetworks are matching when the lines cross. Networks whose IMP subnetworks were matching when rewinding to step 0 (Figure 7, left) generally remain matching at later iterations (except for ResNet-20 low and VGG-16 low at the latest rewinding points). Notably, networks whose IMP subnetworks were *not* matching when rewinding to iteration 0 (Figure 7, right) become matching when rewinding later. Moreover, these rewinding points closely coincide with those where the subnetworks become stable. In summary, at these extreme sparsities, IMP subnetworks are matching when they are stable.

Other observations. Interestingly, the same pattern holds for the train error: for those networks whose IMP subnetworks were not matching at step 0, train error decreases when rewinding later. For ResNet-20 and VGG-16, rewinding makes it possible for the IMP subnetworks to converge to 0% train error. These results suggest stable IMP subnetworks also optimize better.

Randomly pruned and reinitialized subnetworks are unstable and non-matching at all rewinding points (with LeNet again an exception). Although it is beyond the scope of our study, this behavior suggests a potential broader link between subnetwork stability and accuracy: IMP subnetworks are matching and become stable at least as early as the full networks, while other subnetworks are less accurate and unstable for the sparsities and rewinding points we consider.
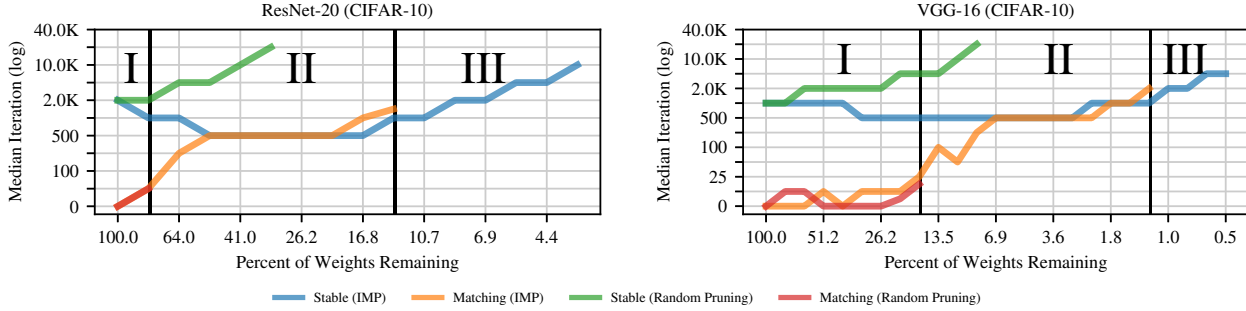
Figure 9. The median rewinding iteration at which IMP subnetworks and randomly pruned subnetworks of ResNet-20 and VGG-16 become stable and matching. A subnetwork is stable if instability $< 2\%$. A subnetwork is matching if the accuracy drop $< 0.2\%$; we only include points where a majority of subnetworks are matching at any rewinding iteration. Each line is the median across three initializations and three data orders (nine samples in total).

4.4. Results at Other Sparsity Levels

Thus far, we have performed instability analysis at only two sparsities: unpruned networks (Section 3) and an extreme sparsity (Section 4.3). Here, we examine sparsities between these levels and beyond for ResNet-20 and VGG-16. Figure 9 presents the median iteration at which IMP and randomly pruned subnetworks become stable (instability $< 2\%$) and matching (accuracy drop $< 0.2\%$, allowing a small margin for noise) across sparsity levels.[2]

Stability behavior. As sparsity increases, the iteration at which the IMP subnetworks become stable decreases, plateaus, and eventually increases. In contrast, the iteration at which randomly pruned subnetworks become stable only increases until the subnetworks are no longer stable at any rewinding iteration.

Matching behavior. We separate the sparsities into three *ranges* where different sparse networks are matching.

In sparsity range I, the networks are so overparameterized that even randomly pruned subnetworks are matching (red). These are sparsities we refer to as *trivial*. This range occurs when more than 80.0% and 16.8% of weights remain for ResNet-20 and VGG-16.

In sparsity range II, the networks are sufficiently sparse that only IMP subnetworks are matching (orange). This range occurs when 80.0%-13.4% and 16.8%-1.2% of weights remain in ResNet-20 and VGG-16. For part of this range, IMP subnetworks become matching and stable at approximately the same rewinding iteration; namely, when 51.2%-13.4% and 6.9%-1.5% of weights remain for ResNet-20 and VGG-16. In Section 4.3, we observed this behavior for a single, extreme sparsity level for each network. Based on Figure 9, we conclude that there are many sparsities where these rewinding iterations coincide for ResNet-20 and VGG-16.

---

[2]In Appendix D, we present the full instability and error data that we used to produce this summary.

In sparsity range III, the networks are so sparse that even IMP subnetworks are not matching at any rewinding iteration we consider. This range occurs when fewer than 13.4% and 1.2% of weights remain for ResNet-20 and VGG-16. According to Appendix D, the error of IMP subnetworks still decreases when they become stable (although not to the point that they are matching), potentially suggesting a broader relationship between instability and accuracy.

5. Discussion

Instability analysis. We introduce instability analysis as a novel way to study the sensitivity of a neural network's optimization trajectory to SGD noise. In doing so, we uncover a class of situations in which *linear* mode connectivity emerges, whereas previous examples of mode connectivity (e.g., between networks trained from different initializations) at similar scales required nonlinear paths (Draxler et al., 2018; Garipov et al., 2018).

Our full network results divide training into two phases: an unstable phase where the network finds linearly unconnected minima due to SGD noise and a stable phase where the linearly connected minimum is determined. Our finding that stability emerges early in training adds to work suggesting that training comprises a noisy first phase and a less stochastic second phase. For example, the Hessian eigenspectrum settles into a few large values and a bulk (Gur-Ari et al., 2018), and large-batch training at high learning rates benefits from learning rate warmup (Goyal et al., 2017).

One way to exploit our findings is to explore changing aspects of optimization (e.g., learning rate schedule or optimizer) similar to Goyal et al. (2017) once the network becomes stable to improve performance; instability analysis can evaluate the consequences of doing so. We also believe instability analysis provides a scientific tool for topics related to the scale and distribution of SGD noise, e.g., the relationship between batch size, learning rate, and general-

ization (LeCun et al., 2012; Keskar et al., 2017; Goyal et al., 2017; Smith & Le, 2018; Smith et al., 2018) and the efficacy of alternative learning rate schedules (Smith, 2017; Smith & Topin, 2018; Li & Arora, 2020).

The lottery ticket hypothesis. The lottery ticket hypothesis (Frankle & Carbin, 2019) conjectures that any "randomly initialized, dense neural network contains a subnetwork that—when trained in isolation—matches the accuracy of the original network." This work is among several recent papers to propose that merely sparsifying at initialization can produce high performance neural networks (Mallya et al., 2018; Zhou et al., 2019; Ramanujan et al., 2020; Evci et al., 2020). Frankle & Carbin support the lottery ticket hypothesis by using IMP to find matching subnetworks at initialization in small vision networks. However, follow-up studies show (Liu et al., 2019; Gale et al., 2019) and we confirm that IMP does not find matching subnetworks at nontrivial sparsities in more challenging settings. We use instability analysis to distinguish the successes and failures of IMP as identified in previous work. In doing so, we make a new connection between the lottery ticket hypothesis and the optimization dynamics of neural networks.

Practical impact of rewinding. By extending IMP with rewinding, we show how to find matching subnetworks in much larger settings than in previous work, albeit from *early* in training rather than initialization. Our technique has already been adopted for practical purposes. Morcos et al. (2019) show that subnetworks found by IMP with rewinding transfer between vision tasks, meaning the effort of finding a subnetworks can be amortized by reusing it many times. Renda et al. (2020) show that IMP with rewinding prunes to state-of-the-art sparsities, matching or exceeding the performance of standard techniques that fine-tune at a low learning rate after pruning (e.g., Han et al., 2015; He et al., 2018). Other efforts use rewinding to further study lottery tickets (Yu et al., 2020; Frankle et al., 2020; Caron et al., 2020; Savarese et al., 2020; Yin et al., 2020).

Pruning. In larger-scale settings, IMP subnetworks only become stable and matching after the full network has been trained for some number of steps. Recent proposals attempt to prune networks at initialization (Lee et al., 2019; Wang et al., 2020), but our results suggest that the best time to do so may be after some training. Likewise, most pruning methods only begin to sparsify networks late in training or after training (Han et al., 2015; Gale et al., 2019; He et al., 2018). The existence of matching subnetworks early in training suggests that there is an unexploited opportunity to prune networks much earlier than current methods.

6. Conclusions

We propose instability analysis as a way to shed light on how SGD noise affects the outcome of optimizing neural networks. We find that standard networks for MNIST, CIFAR-10, and ImageNet become *stable* to SGD noise early in training, after which the outcome of optimization is determined to a linearly connected minimum.

We then apply instability analysis to better understand a key question at the center of the lottery ticket hypothesis: why does iterative magnitude pruning find sparse networks that can train from initialization to full accuracy in smaller-scale settings (e.g., MNIST) but not on more challenging tasks (e.g., ImageNet)? We find that extremely sparse IMP subnetworks only train to full accuracy when they are stable to SGD noise, which occurs at initialization in some settings but only after some amount of training in others.

Instability analysis and our linear mode connectivity criterion contribute to a growing range of empirical tools for studying and understanding the behavior of neural networks in practice. In this paper, we show that it has already yielded new insights into neural network training dynamics and lottery ticket phenomena.

References

Caron, M., Morcos, A., Bojanowski, P., Mairal, J., and Joulin, A. Finding winning tickets with limited (or no) supervision, 2020.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, 2018.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners, 2020.

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

Frankle, J., Schwab, D. J., and Morcos, A. S. The early phase of neural network training. In *International Conference on Learning Representations*, 2020.

Freeman, C. D. and Bruna, J. Topology and geometry of half-rectified network optimization. In *International Conference on Learning Representations*, 2017.

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

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pp. 8789–8798, 2018.

Google. Networks for Imagenet on TPUs, 2018. URL https://github.com/tensorflow/tpu/tree/master/models/.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch SGD: training Imagenet in 1 hour, 2017.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

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

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Lee, N., Ajanthan, T., and Torr, P. H. S. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017.

Li, Z. and Arora, S. An exponential learning rate schedule for deep learning. In *International Conference on Learning Representations*, 2020.

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

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 67–82, 2018.

Morcos, A., Yu, H., Paganini, M., and Tian, Y. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems*, pp. 4932–4942, 2019.

Nagarajan, V. and Kolter, J. Z. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 11615–11626, 2019.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11893–11902, 2020.

Reed, R. Pruning algorithms: A survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

Renda, A., Frankle, J., and Carbin, M. Comparing fine-tuning and rewinding in neural network pruning. In *International Conference on Learning Representations*, 2020.

Savarese, P., Silva, H., and Maire, M. Winning the lottery with continuous sparsification. *arXiv preprint arXiv:1912.04427*, 2020.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of residual networks using large learning rates. In *International Conference on Learning Representations*, 2018.

Smith, S. L. and Le, Q. V. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

Smith, S. L., Kindermans, P.-J., and Le, Q. V. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

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

Yin, S., Kim, K.-H., Oh, J., Wang, N., Serrano, M., Seo, J.-S., and Choi, J. The sooner the better: Investigating structure of early winning lottery tickets, 2020.

Yu, H., Edunov, S., Tian, Y., and Morcos, A. S. Playing the lottery with rewards and multiple languages: Lottery tickets in rl and nlp. In *International Conference on Learning Representations*, 2020.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In *Advances in Neural Information Processing Systems*, pp. 3592–3602, 2019.