MODE CONNECTIVITY AND SPARSE NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We uncover a connection between two seemingly unrelated empirical phenomena: *mode connectivity* and *sparsity*. On the one hand, there is growing catalog of situations where, across multiple runs, SGD learns weights that fall into minima that are connected (*mode connectivity*). A striking example is described by Nagarajan & Kolter (2019). They observe that test error on MNIST does not change along the *linear* path connecting the end points of two independent SGD runs, starting from the same random initialization. On the other hand, there is the lottery ticket hypothesis of Frankle & Carbin (2019), where dense, randomly initialized networks have sparse subnetworks capable of training in isolation to full accuracy. However, neither phenomenon scales beyond small vision networks.

We start by proposing a technique to find sparse subnetworks *after initialization*. We observe that these subnetworks match the accuracy of the full network only when two SGD runs for the same subnetwork are connected by linear paths with the no change in test error. Our findings connect the existence of sparse subnetworks that train to high accuracy with the dynamics of optimization via mode connectivity. In doing so, we identify analogues of the phenomena uncovered by Nagarajan & Kolter and Frankle & Carbin in ImageNet-scale architectures at state-of-the-art sparsity levels.

1 INTRODUCTION

Our understanding of training large neural networks via stochastic gradient descent (SGD) is evolving rapidly, in large part due to work that uncovers fascinating empirical phenomena. One such set of phenomena relate to *mode connectivity*: the observation that SGD solutions are connected through paths of approximately equal loss (Draxler et al., 2018; Garipov et al., 2018; Kuditipudi et al., 2019; Venturi et al., 2018; Liang et al., 2018; Nguyen et al., 2018; Nguyen, 2019). In recent work, Nagarajan & Kolter (2019) observe that, after training two fully connected networks on disjoint subsets of MNIST from the same initialization, the final weights are connected by a linear path of constant test error. This particularly striking example of mode connectivity—where SGD seems to find the same local *convex* sublevel set—appears to *not* reliably replicate in larger networks (see Appendix C).

Another interesting set of phenomena relate to the *Lottery Ticket Hypothesis* (Frankle & Carbin, 2019). Using a technique based on *iterative magnitude pruning* (IMP), Frankle & Carbin find that dense networks contain sparse subnetworks that can be trained in isolation from the original initialization to match the accuracy of the original dense network. At less extreme levels of sparsity, reusing the initialized weights becomes less important (Liu et al., 2019; Lee et al., 2019). Several experiments (Liu et al., 2019; Gale et al., 2019), including some in the original paper, show that the "lottery ticket" phenomenon does not replicate in larger networks.

Although mode connectivity and lottery tickets are seemingly unrelated phenomena, we uncover a relationship in this work. The starting point is an observation about the behavior of IMP subtnetworks of larger networks. While these subnetworks may fail to match the accuracy of the full network when trained from their original initializations, we find that they often do when trained from their weights *rewound* to some training iteration $k > 0$. We observe this behavior extremely

Algorithm 1 Iterative Magnitude Pruning (IMP) with rewinding to iteration k and N iterations.

1: Randomly initialize a neural network $f(x; m \odot W_0)$ with initial trivial pruning mask $m = 1^{|W_0|}$.
2: Train the network for k iterations, producing network $f(x; m \odot W_k)$.
3: **for** $n \in \{1, \ldots, N\}$ **do**
4: Train $f(x; m \odot W_k)$ for $T - k$ iterations, producing network $f(x; m \odot W_T)$.
5: Prune the remaining entries with the lowest magnitudes from W_T. Let $m[i] = 0$ if $W_T[i]$ is pruned.
6: Return $f(x; m \odot W_k)$

early in training (less than 5% of the way through, and sometimes 1% or less)[1] on standard vision networks for CIFAR10 and ImageNet at sparsity levels typically obtained by state-of-the-art pruning algorithms (Gale et al., 2019; He et al., 2018).

The relationship between mode connectivity and lottery tickets concerns the point during training when IMP first starts to pick out subnetworks that train to full accuracy (*matching subnetworks*): At the earliest rewinding iteration k_0 that leads to a matching subnetwork, subnetworks produced by IMP exhibit the same behavior observed by Nagarajan & Kolter (2019): if we train these subnetworks from iteration k_0 to completion *on different data orders*, we find that linear interpolations between the end points have the same test error. In contrast, interpolation leads to excess test error for IMP subnetworks and randomly pruned subnetworks that train to lower accuracy. In short, in this sparse regime, a subnetwork is matching if and only if it is stable. Our observations identify an analogue of the mode connectivity behavior found by Nagarajan & Kolter (2019), but for large networks for ImageNet, and connect this behavior to lottery ticket phenomena.

We call this particular form of mode connectivity *stability*: a subnetwork at iteration k is stable if it is possible to linearly interpolate between two *duplicates* trained on different data orders with no increase in test error. Remarkably, stability alone explains the known successes and failures of the lottery ticket hypothesis: IMP subnetworks at initialization are matching subnetworks only when they are stable, and hyperparameter changes that Frankle & Carbin make to find matching subnetworks at initialization in more challenging settings also lead to stability (Section 3). Rewinding IMP to an iteration just after initialization makes it possible to find stable, matching subnetworks in cases that Frankle & Carbin found impossible (Section 4) and on ImageNet-scale problems (Section 5). We close with a discussion of the implications of these observations for other research (Section 6).

Contributions.

- We extend iterative magnitude pruning (IMP) to *rewind* weights to an arbitrary iteration $k \geq 0$.
- We observe that, for a suitable choice of rewinding iteration, IMP identifies (*matching subnetworks*): sparse subnetworks that train to full accuracy. In the networks we study, this iteration is always early in training, but often after initialization for larger networks.
- We introduce the notion of a *stable* subnetworks: those where SGD reliably finds the same local convex sublevel set during training, despite the stochasticity in the minibatch data order.
- Focusing on cases where some rewinding iteration exists that leads to a matching subnetwork, we find that sparse subnetworks become matching at the iteration where they become stable.
- We use these ideas to find sparse matching subnetworks of networks for CIFAR10 and ImageNet.

2 Preliminaries and Methodology

Throughout this paper, we investigate sparse subnetworks of dense neural networks. Formally, a *subnetwork* is a tuple (W, m) of weights $W \in \mathbb{R}^D$ and a mask $m \in \{0, 1\}^D$. We write $m \odot W$ to denote the element-wise product of a mask with weights. We capture the effect of training with stochastic gradient descent (or some variant) by a function $\mathcal{A}^{s \to t} : \mathbb{R}^D \times U \to \mathbb{R}^D$, which maps weights W_s at iteration s and data order randomness $u \sim U$ to updated weights W_t at iteration t for $s, t \in \{1, .., T\}$ and $s < t$. Let $\mathcal{E}(W, m)$ denote the error of subnetwork (W, m). In practice, we use a held-out test set to estimate this quantity.

[1] Naively, one would assume that rewinding to a point late in training, after optimization is nearly complete, would lead to this behavior. However, pruning at this point reliably leads to a drop in performance that requires many epochs of *fine-tuning* to recover.

Network	Dataset	Params	Train For	Batch	Accuracy	Opt	Rate	Schedule	Warmup	Density
Lenet	MNIST	266K	50K Iters	60	$98.1 \pm 0.2\%$	adam	12e-4	constant	0	2.8%
Resnet-20 (standard)	CIFAR10	274K	30K Iters	128	$91.0 \pm 0.3\%$	mom.	0.1	10x drop at 20K, 25K	0	16.7%
Resnet-20 (low)					$89.1 \pm 0.2\%$	mom.	0.01		0	
Resnet-20 (warmup)					$90.1 \pm 0.2\%$	mom.	0.03		20K	
VGG-19 (standard)	CIFAR10	20.0M	112K Iters	64	$91.4 \pm 0.3\%$	mom.	0.1	10x drop at 56K, 84K	0	2.2%
VGG-19 (low)					$91.8 \pm 0.2\%$	mom.	0.01		0	
VGG-19 (warmup)					$92.6 \pm 0.1\%$	mom.	0.1		10K	
Resnet-50	ImageNet	25.5M	90 Eps	1024	$76.1 \pm 0.1\%$	mom.	0.4	10x drop at 30,60,80	5 Eps	30%
Inception-v3	ImageNet	27.1M	171 Eps	1024	$78.1 \pm 0.1\%$	mom.	0.03	linear decay to 0.005	0	30%

Table 1: Networks and hyperparameters for our experiments. Accuracies are the averages and standard deviations across three initializations and three data orders (18 runs total). Hyperparameters for Resnet-20 (standard) are from He et al. (2016) via Frankle & Carbin (2019). Hyperparameters for VGG-19 (standard) are from Liu et al. (2019). Hyperparameters for *low* and *warmup* variants and Lenet are from Frankle & Carbin (2019). Hyperparameters for ImageNet networks are from Google's reference TPU library (Google, 2018). MNIST and CIFAR10 networks are identical to those used in Frankle & Carbin (2019).

Iterative magnitude pruning. To find sparse networks early in training, we build off of a procedure proposed by Frankle & Carbin (2019), which we term *Iterative Magnitude Pruning* (IMP). As outlined in Algorithm 1 with $k = 0$, IMP trains a network to completion, prunes weights with the lowest magnitudes globally, and *rewinds* the remaining weights back to their initial values. Doing so produces a subnetwork of the original network. We say that a subnetwork is *matching* when training that network to completion leads to accuracy that matches that of the trained, unpruned network. When IMP picks out a subnetwork that is matching, we deem IMP to have *succeeded*. We run IMP iteratively, training, pruning 20% of weights (Han et al., 2015; Frankle & Carbin, 2019), rewinding, and repeating until we reach a target sparsity.

In Sections 4 and 5, we search for matching subnetworks after initialization. To do so, we generalize IMP to rewind the weights to some *rewinding iteration* k rather than just to iteration 0. Algorithm 1 describes this generalization. When training a subnetwork from iteration k, we set the learning rate schedule to iteration k and complete the training process normally from that point.

Stability. We measure the stability of a subnetwork by first training two *duplicates* of the subnetwork in an identical fashion but for different minibatch data orders. Subsequently, we look at weights obtained by linearly interpolating between the two trained networks and measure the increase in worst-case increase in test error over all interpolations. (We approximate the supremum over a grid of 50 points.) We say the subnetwork is ϵ-stable if the maximum increase in test error is ϵ. If test error does not increase as we linearly interpolate, then this provides evidence that the duplicates may occupy the same local (convex) sublevel set (local optimum) and we call the subnetwork *stable*. If test error increases, then we can come to no conclusion about whether the duplicates occupy the same local optimum.

We can view the computation outlined above as a statistical estimate of the following notion of stability: Consider a subnetwork (W_t, m) and let $(W_T^u, m) = \mathcal{A}^{t \to T}(m \odot W_t, u)$ be the subnetwork trained to completion for some random data order. Then $\bar{\mathcal{E}} = \mathbb{E}_{u \sim U}[\mathcal{E}(W_T^u, m)]$ is the expected error of the final weights, averaging over the data order. We are then estimating $\mathbb{E}_{u,u' \sim U}[\sup_{a \in [0,1]} \mathcal{E}(aW_T^u + (1-a)W_T^{u'}, m) - \bar{\mathcal{E}}]$ and so we say a subnetwork (W_t, m) is ϵ-*stable* if this quantity is bounded above by ϵ. A *stable* network is one that is 0-stable.

Networks and datasets. We study image classification networks on MNIST, CIFAR10, and ImageNet as specified in Table 1. All hyperparameters listed are the standard values for these networks from reference implementations or prior work as cited in Table 1. The *warmup* and *low* variants of Resnet-20[2] and VGG-19 use the learning rate schedules chosen in Frankle & Carbin (2019) to make it possible for IMP to find matching subnetworks at iteration 0.

[2]Frankle & Carbin (2019) mistakenly refer to Resnet-20 for CIFAR10 as "Resnet-18," which is a separate network architecture designed for ImageNet.

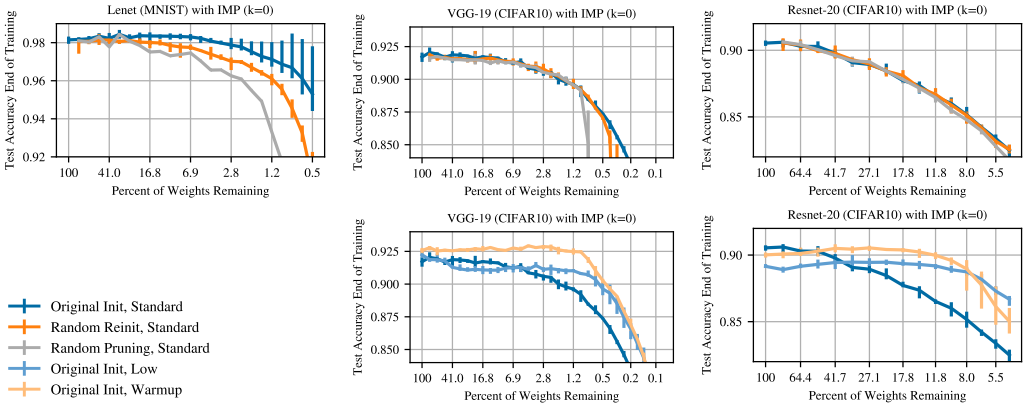


Figure 1: The accuracy of subnetworks of Lenet (left), VGG-19 (middle), and Resnet-20 (right) found using IMP (k=0) and when randomly reinitialized and randomly pruned. Standard, Warmup, and Low hyperparameters are in Table 1. Error bars are the min/max across five runs.

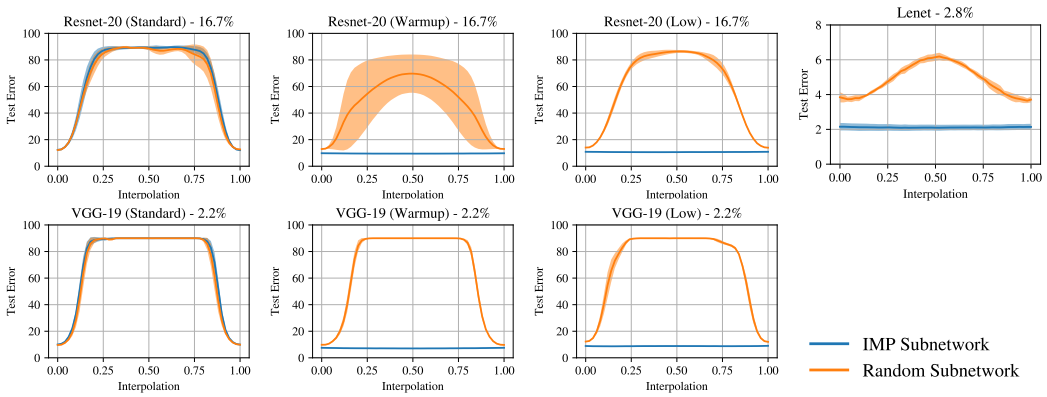


Figure 2: The test error when interpolating at 50 points between IMP subnetworks (blue) and random subnetworks (orange) trained on different data orders. Error bars are the standard deviation across 18 combinations of networks (three data orders and three initializations per network). The subnetworks are pruned to the sparsity levels in Table 1.

Sparsity levels. We study the relationship between the stability and accuracy of sparse subnetworks. We only consider subnetworks within a restricted sparsity range: in particular, we focus on sparsity levels for which we have evidence that there exists a matching subnetwork at some rewinding iteration $k \geq 0$. Furthermore, we focus our attention to the lowest few sparsity levels at which IMP can find a matching subnetwork for some rewinding iteration. These sparsity levels are noted in Table 1.

3 Matching Subnetworks at Initialization are Stable

In this section, we investigate subnetworks found by IMP at initialization in order to compare scenarios where IMP succeeds and fails at finding matching subnetworks. For the sparsity levels we consider, we find matching subnetworks if and only if they are stable.

IMP at initialization. On Lenet for MNIST, IMP finds matching subnetworks at initialization (Figure 1 left, blue) when at least 2.8% of weights remain. The original initialization and structure are necessary to produce this behavior: reinitializing the subnetwork (orange) or randomly pruning[3] (gray) reduce subnetwork accuracy at this sparsity level and only result in matching subnetworks when at least 20.1% of weights remain.

However, IMP does not find matching subnetworks on VGG-19 and Resnet-20 at initialization at even moderate levels of sparsity (Figure 1 top middle and right, blue). The original initialization is

[3]We generate a random mask with the same layer-wise proportions as the corresponding IMP subnetwork.

Network	Stability (ϵ)		Accuracy			
	IMP	Random	IMP	Random	Full - IMP	Matching?
Lenet	0.1 ± 0.0	2.4 ± 0.5	97.9 ± 0.2	96.3 ± 0.2	0.3 ± 0.3	Y
Resnet-20 (standard)	78.1 ± 0.9	77.4 ± 1.3	87.7 ± 0.4	87.7 ± 0.3	3.2 ± 0.3	N
Resnet-20 (low)	0.1 ± 0.1	72.5 ± 1.3	89.1 ± 0.2	86.0 ± 0.3	-0.1 ± 0.4	Y
Resnet-20 (warmup)	0.1 ± 0.1	56.8 ± 20.2	90.2 ± 0.3	86.8 ± 0.2	0.1 ± 0.4	Y
VGG-19 (standard)	80.0 ± 0.3	80.2 ± 0.3	90.0 ± 0.2	90.2 ± 0.2	1.4 ± 0.4	N
VGG-19 (low)	0.2 ± 0.2	77.9 ± 0.3	91.0 ± 0.2	88.0 ± 0.2	0.8 ± 0.2	N
VGG-19 (warmup)	0.1 ± 0.1	80.2 ± 0.3	92.4 ± 0.2	90.3 ± 0.1	0.1 ± 0.2	Y

Table 2: The minimum ϵ for stability and the accuracy of subnetworks obtained by IMP and by randomly pruning. Errors are the standard deviation across three initializations and data orders, meaning we considered three networks for accuracy and 18 combinations of networks for stability.

inconsequential, and the networks can be reinitialized or randomly pruned without affecting accuracy. Only by changing the learning schedule do Frankle & Carbin (2019) find matching subnetworks (Figure 1 bottom), lowering the learning rate (light blue) or adding warmup (light orange).

Stability at initialization. To compute stability, we train two duplicates of a subnetwork with different data orders and linearly interpolate between the trained weights (Section 2). The blue lines in Figure 2 plot the test error when linearly interpolating between duplicates of the IMP subnetworks. Nagarajan & Kolter (2019) study this phenomenon on a Lenet-like network for MNIST, and we also confirm their result on sparse IMP subnetworks: it is possible to linearly interpolate between minima with no increase in test error.

We observe this behavior on several other networks on CIFAR. Namely, whenever IMP finds a matching[4] subnetwork, test error does not increase when linearly interpolating between duplicates, meaning the subnetwork is stable. This result extends Nagarajan & Kolter's observation about linear interpolation beyond MNIST to matching subnetworks found by IMP at initialization on our CIFAR10 networks.[5]

In contrast, when IMP does not find a matching subnetwork, test error increases when interpolating, meaning the subnetworks are not linearly connected within an optimum and are not stable. As a baseline, we also consider randomly pruned networks (orange line). These subnetworks are only matching at moderate levels of sparsity (Figure 1); correspondingly, at the sparsity levels we consider, we cannot interpolate between duplicates without increases in test error.

Table 2 reports the level of stability from the curves in Figure 2.[6] In cases where we find matching subnetworks, we also find that $\epsilon \approx 0$, meaning the subnetworks are stable; in cases where we do not find matching subnetworks, we find $\epsilon > 0$, meaning the subnetworks are unstable.[7] As additional evidence for a relationship between stability and matching subnetworks, the interventions that Frankle & Carbin took to improve the accuracy of IMP subnetworks (warmup and/or low) also result in stability when subnetworks would otherwise be unstable.

[4] A subnetwork is considered matching when the mean difference in accuracy from the full network is smaller than the standard deviation.

[5] In Appendix C, we find that, aside from Lenet, the corresponding full networks are *not* stable at initialization, despite the fact that the IMP subnetworks are stable.

[6] Concretely, the value is the maximum point on the curve (i.e., maximum risk along the linear interpolation), minus the mean of the endpoints, averaged across trials.

[7] Following Frankle & Carbin's iterative pruning strategy prescribes a fixed set of sparsity levels (Section 2) and for the sparsity levels we studied on VGG (low), the IMP subnetwork is stable but does not quite qualify as matching. In Section 4, we find that VGG-19 (low) requires just ten iterations of training to become matching, meaning that, at initialization, it is on the cusp of matching. At initialization, IMP finds subnetworks of VGG-19 (low) that reach higher much accuracy than the random baselines at extreme levels of sparsity, however, accuracy does not reach that of the full network.

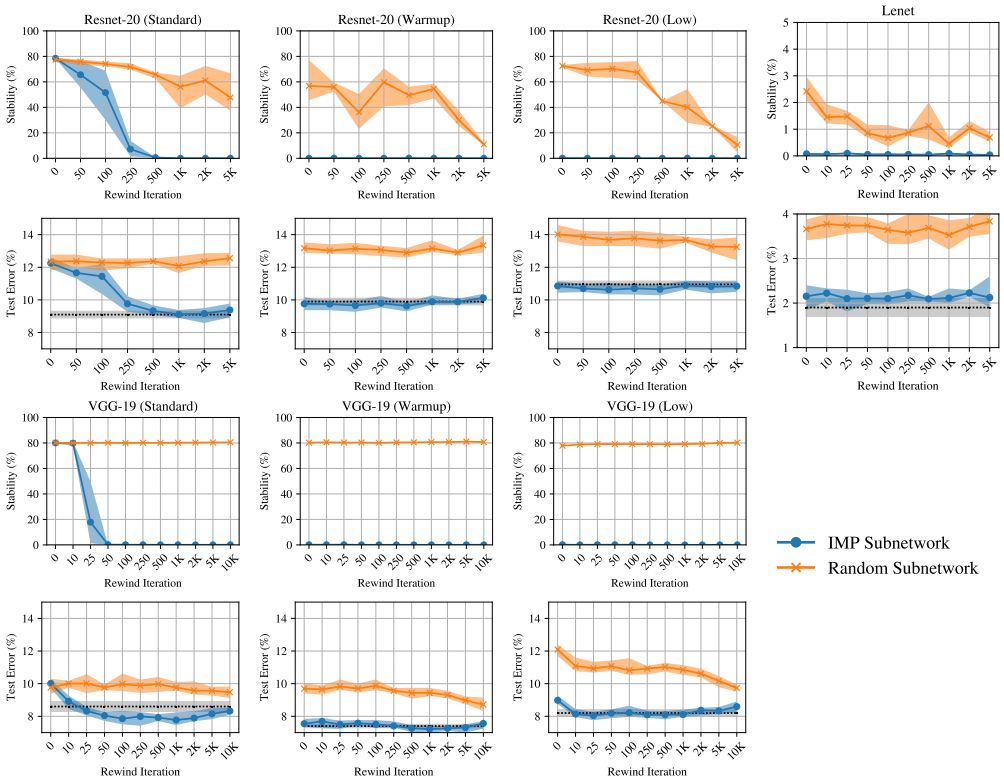


Figure 3: The effect of the rewinding iteration (x-axis) on stability (top row) and test error (bottom row) for the MNIST and CIFAR10 networks in Table 1. Error bars are the standard deviations across three initializations (and three data orders for the top row). Dashed black lines are the mean and standard deviation of test error of the full network from Figure 1.

4 Stability and Accuracy Rapidly Improve After Initialization

In this section, we study stability and accuracy when rewinding IMP subnetworks to iterations after initialization. For networks that were not already stable at initialization, stability and accuracy improve rapidly as we rewind to later iterations, culminating in stable, matching subnetworks.

Stability after initialization. To test whether subnetworks become stable later in training, we use IMP and random pruning to generate subnetworks at various *rewinding* iterations k after initialization. We train these subnetworks from iteration k to completion according to the learning schedule. The upper plot in Figure 3 shows stability for IMP subnetworks (blue) and randomly pruned subnetworks (orange) across rewinding iterations.

For Resnet-20 (standard) and VGG-19 (standard), stability of IMP subnetworks improves as the rewinding iteration increases. This change takes place early on, culminating with stable subnetworks at iteration 100 (epoch 0.14) for VGG-19 and iteration 500 (epoch 1.4) for Resnet-20. After this point, stability improvements largely saturate. In cases where IMP finds a matching subnetwork at iteration 0 (Lenet, Resnet-20 Warmup, Resnet-20 Low, and VGG-19 Warmup), IMP subnetworks are already stable and remain so at all rewinding iterations.

Randomly pruned subnetworks are unstable over the range of rewinding iterations we consider, although they become somewhat more stable with later rewinding for Lenet and Resnet-20. One possible explanation for why the test error of these random subnetworks is high for later rewinding iterations despite a slight improvement in stability is the following: by the time these improvements manifest, Lenet has already converged and Resnet-20 is 20% of the way through training, sufficiently late that the subnetworks may not have enough time to venture as far from their weights at iteration k (see L2 distance measurements in Appendix A.3).

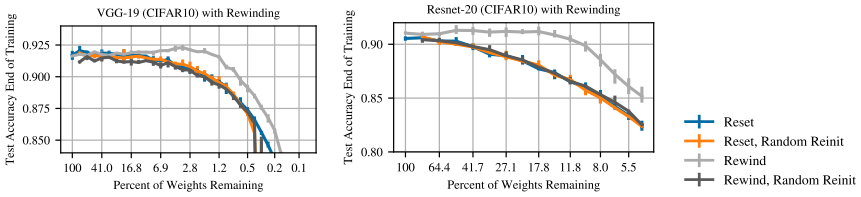


Figure 4: The accuracy of subnetworks of VGG-19 Standard (left) and Resnet-20 Standard (right) found using IMP with resetting, rewinding (iterations 100 and 500 for VGG-19 and Resnet-20), and when randomly reinitialized. Error bars are the min/max across five runs.

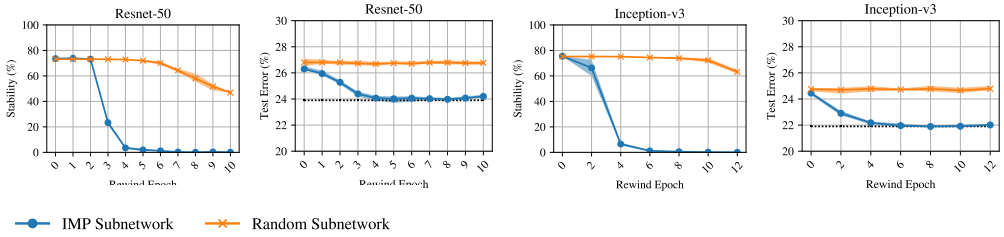


Figure 5: The effect of the rewinding epoch (x-axis) on stability (left) and accuracy (right) for IMP subnetworks of networks and datasets in Table 1 trained on ImageNet. Sparsity levels are in Table 1. Subnetworks were obtained via one-shot pruning. Error bars are the standard deviation across three comparisons (three initializations and two data orders per initialization).

These results are another extension of the phenomenon originally observed by Nagarajan & Kolter (2019) on Lenet. For all of the networks we study, there are constant-error linear paths connecting the optima of IMP subnetworks at sufficiently late rewinding iterations. In Appendix C, we find such linear paths when rewinding the full, unpruned networks as well, suggesting that instability during the early part of training may be an intrinsic property of SGD on these networks.

Accuracy after initialization. In Section 3, we found that stability and accuracy improve together: interventions intended to improve accuracy (namely, a lower learning rate and warmup) also improve stability. Here, we find that the same relationship holds: the rewinding that makes subnetworks stable (Figure 3 top) also improves accuracy (Figure 3 bottom).

For Resnet-20 (standard) and VGG-19 (standard), error decreases as stability improves, and improvements in error saturate when IMP subnetworks become stable. Although rewinding to iteration 0 does not produce matching subnetworks, rewinding slightly later (iteration 500 out of 30K for Resnet-20 and 100 out of 112K for VGG-19) does. For other networks, which are stable at rewinding iteration 0 and remain so at other rewinding iterations, accuracy remains consistent. For VGG-19 low and warmup, random subnetwork accuracy also increases slightly at the latest rewinding points.

Discussion. At the sparsity levels we consider, we continue to see that subnetworks are matching if and only if they are stable. In those cases for which IMP does not find matching subnetworks at initialization, it finds them when rewinding a small way into training. Rewinding has limited effect on the stability and accuracy of randomly pruned subnetworks until the latest rewinding points.

Rewinding improves the accuracy of IMP subnetworks at all sparsity levels. Figure 4 plots the accuracy of Resnet-20 and VGG-19 across sparsity levels when IMP rewinds to iterations 100 (VGG-19) and 500 (Resnet-20). At extreme levels of sparsity, these subnetworks substantially outperform subnetworks that are randomly reinitialized or rewound to initialization.

5 Stability and Accuracy of Networks on ImageNet

In this section, we find the same relationship between stability and accuracy on deeper networks for ImageNet (Russakovsky et al., 2015) (see Table 1). Although IMP with $k = 0$ does not produce matching subnetworks, rewinding 6.6% and 3.5% into training yields subnetworks that are 80% and 76% smaller than the Resnet-50 (He et al., 2016) and Inception-v3 (Szegedy et al., 2016) architectures, respectively, that can complete training to full accuracy.

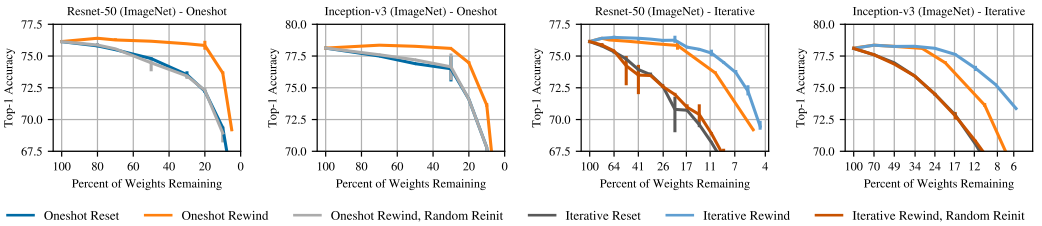


Figure 6: The accuracy of subnetworks of Resnet-50 and Inception-v3 using IMP with rewinding to epochs 0 and 6 and when randomly reinitialized. Left plots: one-shot pruning. Right plots: iteratively pruning 20% and 30% of weights per iteration for Resnet-50 and Inception-v3. Error bars are the min/max across three runs.

Figure 5 shows the effect of the rewinding epoch on the stability and accuracy at sparsity levels in Table 1. The trends mirror those of Resnet-20 and VGG-19. When rewinding to initialization, IMP subnetworks perform no better than random subnetworks in terms of both stability and accuracy. As the rewinding iteration increases, IMP subnetwork stability improves until the subnetworks become stable at epoch 4 (out of 90) for Resnet-50 and epoch 6 (out of 171) for Inception-v3. Test error improves alongside stability, reaching the level of the original network at the same point in training.

Figure 6 shows IMP with rewinding across all levels of sparsity. The left plots shows the result of *one-shot pruning* (pruning all at once to the desired sparsity after training) when rewinding to epoch 6 (just after the aforementioned thresholds) Both networks match the accuracy of the original network when 70% of weights are pruned. The right plots shows iterative pruning. Rewinding Resnet-50 and Inception-v3 to epoch 6 makes it possible to find subnetworks that train to match the original accuracy when 80% and 76% of weights are pruned, matching state-of-the-art sparsity levels (Gale et al., 2019; He et al., 2018) while imposing the sparsity just a few epochs into training. Under any amount of pruning, subnetworks that are rewound to iteration 0 or randomly reinitialized have lower accuracy than the full network.

6 Discussion and Implications

Interpolation between trained networks. Earlier work has contributed to a catalog of scenarios where neural network minima are connected by "simple" (if non-linear) paths of low empirical risk (Draxler et al., 2018; Garipov et al., 2018; Kuditipudi et al., 2019; Venturi et al., 2018; Liang et al., 2018; Nguyen et al., 2018; Nguyen, 2019). A related phenomena is that averaging the states of a model trained with a cyclic learning rate leads to better generalization Izmailov et al. (2018). Another related phenomenon was uncovered by Evci et al. (2019): they show that linear paths between a subnetwork's mode (W_T, m) and its random reinitialization exhibit monotonically decreasing empirical risk, where the subnetwork is obtained via one iteration of IMP. Despite the existence of this linear path, SGD does not seem to follow it.

We extend the technique of Nagarajan & Kolter (2019), who study interpolation of networks trained from the same initialization on disjoint subsets of MNIST. We consider more challenging tasks, sparse subnetworks, and later rewinding points; we also use the full dataset with different data orders. Our findings shed light on the optimization dynamics of sparse (and dense—Appendix C) neural networks under SGD noise. Despite the small number of parameters in matching subnetworks, we find that modes are connected via linear paths. This is surprising, as most of the previous theoretical work on mode connectivity relies on high parameter count or insensitivity to dropout noise (Venturi et al., 2018; Liang et al., 2018; Nguyen et al., 2018; Nguyen, 2019; Kuditipudi et al., 2019). Studying non-linear paths (Draxler et al., 2018; Evci et al., 2019) may reveal other relationships, e.g., between the minima of full networks and subnetworks.

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." The authors find support for this hypothesis in small vision networks using IMP to identify the subnetworks. As follow-up studies show (Liu et al., 2019; Gale et al., 2019), IMP does not uncover such subnetworks on deeper networks

for CIFAR10 and ImageNet. We replicate these findings, and, using the notion of stability, provide an alternative perspective: at high levels of sparsity, a subnetwork is matching if it is stable. For the networks studied by Frankle & Carbin (2019), the subnetworks picked out by IMP are stable at initialization. For those counterexamples raised by subsequent studies, the subnetworks picked out by IMP are not stable at initialization. In our study, we find that the point where stability arises occurs later as networks become larger.

The relationship between stability and accuracy. The connection between stability and matching subnetworks suggests that the accuracy of matching subnetworks relates to the dynamics of optimization. Stability is a precise and surprising fact about the distribution of the weights learned by SGD. Outside of our IMP experimental setup, one might not expect stability to align with accuracy after training. For example, a subnetwork could be trapped in a bad local minimum and be stable. It is an open question as to *why* stable networks are also matching in the setting we study. One could imagine several possible explanations. In Appendix B, we cast doubt on the idea that IMP subnetworks reach the same minimum as the full network: we find matching subnetworks in many cases where there is no low-error linear path between the optima of the full network and the IMP subnetwork.

Opportunities to prune. In Sections 4 and 5, we find sparse subnetworks that become matching early on in training. These subnetworks match or exceed sparsity levels achieved by state-of-the-art (unstructured) pruning methods. Most of these techniques work by pruning during or after training (Gale et al., 2019; He et al., 2018). In these cases, our work suggests that there is, potentially, a substantial unexploited opportunity to prune neural networks much earlier in training. (See Appendix D for more details.) Recent work proposes techniques to prune at initialization (Lee et al., 2019); our experiments suggest that the best time to prune may actually be slightly later, corresponding to when IMP subnetworks become stable.

7 Limitations

We focus only on image classification tasks and sparse pruning. For each neural network, we analyze stability for a single sparsity level intended create matching subnetworks that are as small as possible; other relationships between subnetwork accuracy and stability may emerge in other sparsity regimes. We only consider linear interpolation between optima, which may exclude other simple paths between the optima of other combinations of networks.

IMP is our only technique for finding matching subnetworks, and our reliance on IMP limits the scope of our claims. For example, there may exist other matching subnetworks with a different relationship with stability. It may even be that stability is a necessary condition for IMP to find matching subnetworks rather than a necessary condition for a subnetwork to be matching.

References

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, 2018.

Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks, 2019. arXiv:1906.10732.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Int. Conf. Represent. Learn.*, 2019.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 2019. arXiv:1902.09574.

T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. 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/.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2018. arXiv:1803.05407.

Rohith Kuditipudi, Xiang Wang, Holden Lee, Yi Zhang, Zhiyuan Li, Wei Hu, Sanjeev Arora, and Rong Ge. Explaining landscape connectivity of low-cost solutions for multilayer nets, 2019. arXiv:1906.06247.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. In *International Conference on Learning Representations*, 2019.

Shiyu Liang, Ruoyu Sun, Yixuan Li, and Rayadurgam Srikant. Understanding the loss surface of neural networks for binary classification, 2018. arXiv:1803.00909.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the Value of Network Pruning. In *International Conference on Learning Representations*, 2019.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning, 2019. arXiv:1902.04742v2. To appear in NeurIPS 2019.

Quynh Nguyen. On connected sublevel sets in deep learning, 2019. arXiv:1901.07417.

Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys, 2018. arXiv:1809.10749.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

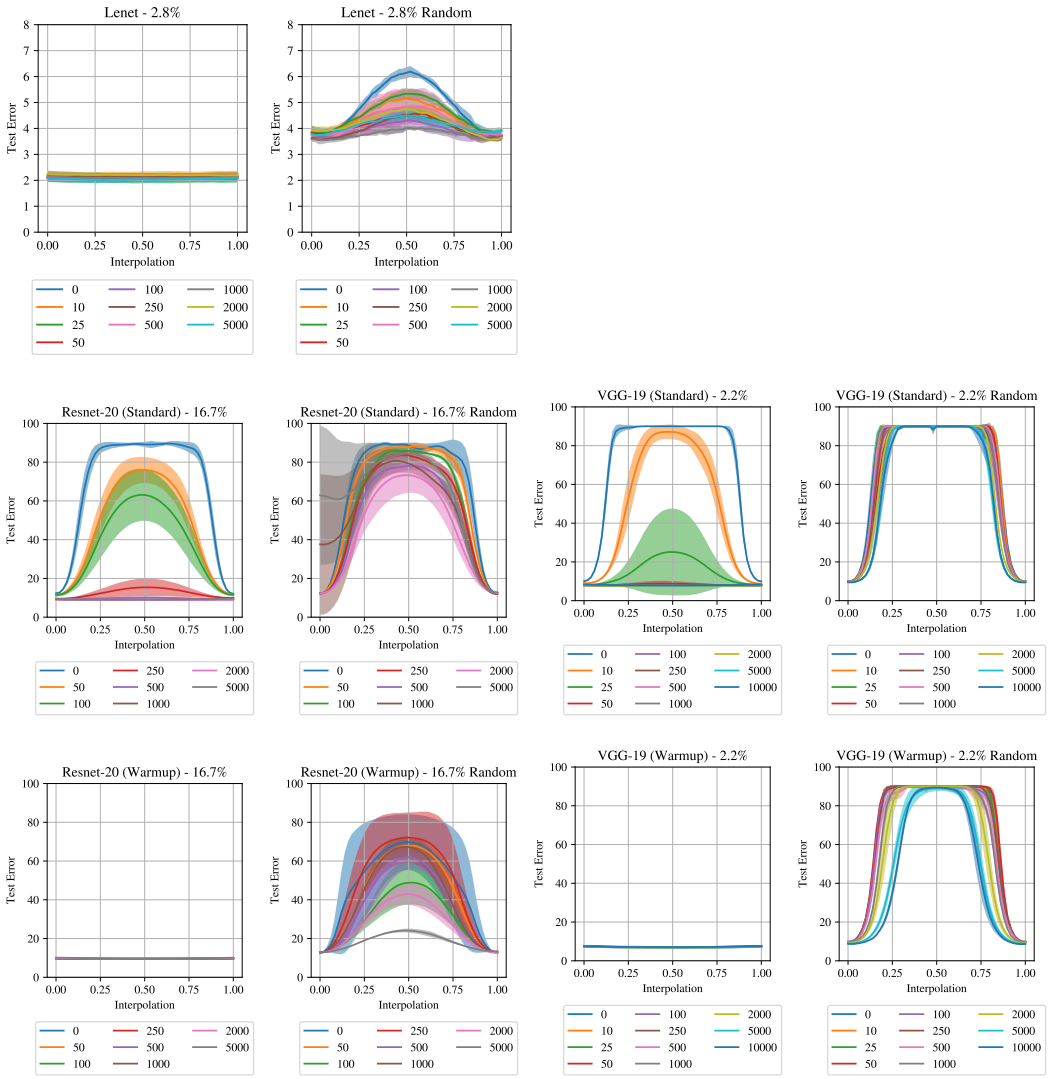
Luca Venturi, Afonso S Bandeira, and Joan Bruna. Spurious valleys in two-layer neural network optimization landscapes, 2018. arXiv:1802.06384.

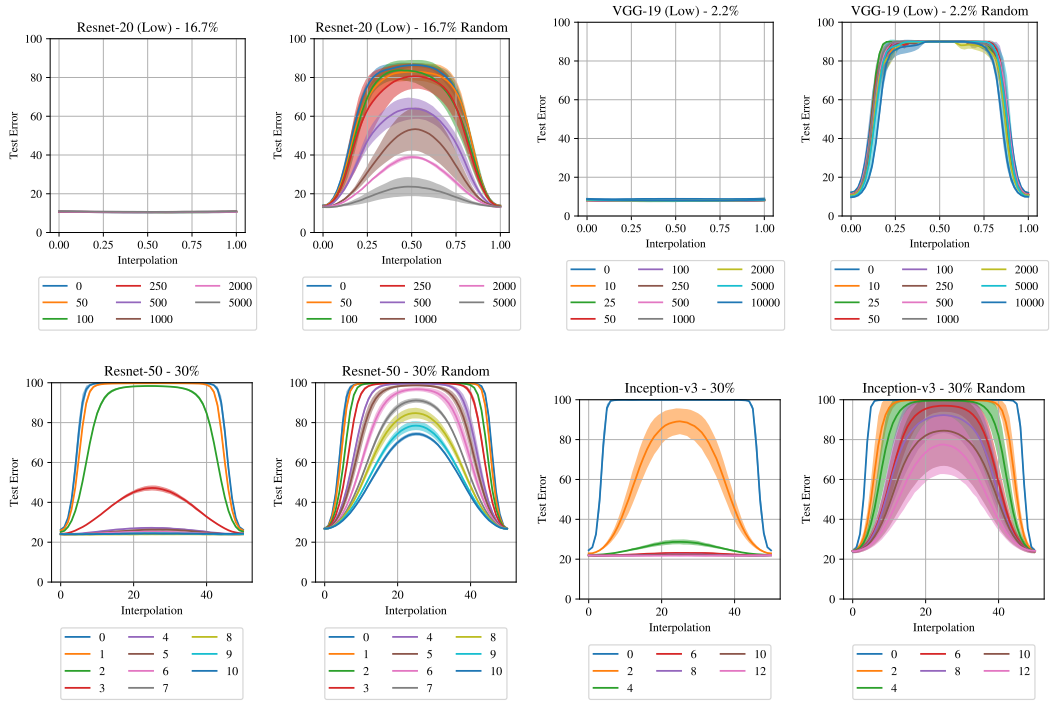
A COMPARING SUBNETWORKS TRAINED WITH DIFFERENT DATA ORDERS

This appendix includes full data for our stability experiments, including plots at all points along the interpolation between subnetworks (Subsection A.1), the interpolation summary plots from the main body of the paper (Subsection A.2), the L_2 distance between pairs of subnetworks trained on different data orders (Subsection A.3), and the angle between pairs of subnetworks trained on different data orders (Subsection A.4). All lines for MNIST and CIFAR10 networks are the average and standard deviation across 18 comparisons (three initializations and three data orders per initialization). All lines for ImageNet networks are the average and standard deviation across three comparisons (three initializations and two data orders per initialization).

A.1 FULL INTERPOLATION PLOTS

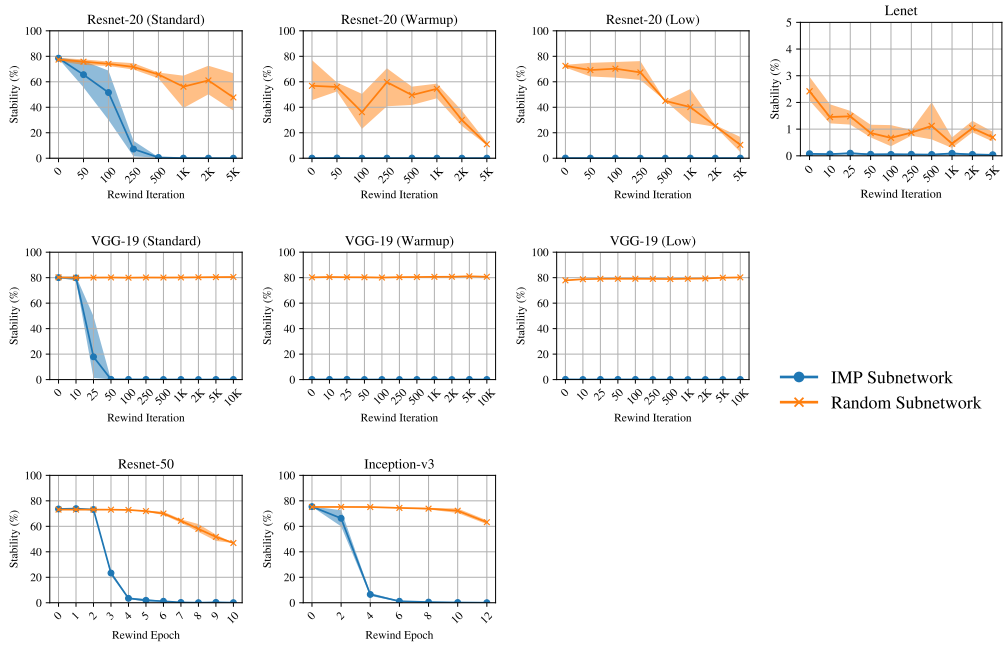
Figures 3 and 5 summarize the stability for each network. This subsection plots the values of the test loss at all 50 points at which we linearly interpolate between each pair of duplicates. For the summary plots in Figures 3 and 5, we select the maximum test error at any point in each curve and subtract from it the mean of the two endpoints of each curve (i.e., the test errors of the uninterpolated networks). For each network, the left graph is the IMP subnetwork and the right graph is the random subnetwork. The fraction of weights remaining is listed in the title. For MNIST and CIFAR10 networks, the numbers are the rewinding iteration; for ImageNet networks, they are the rewinding epoch.





A.2 Interpolation Summary Plots

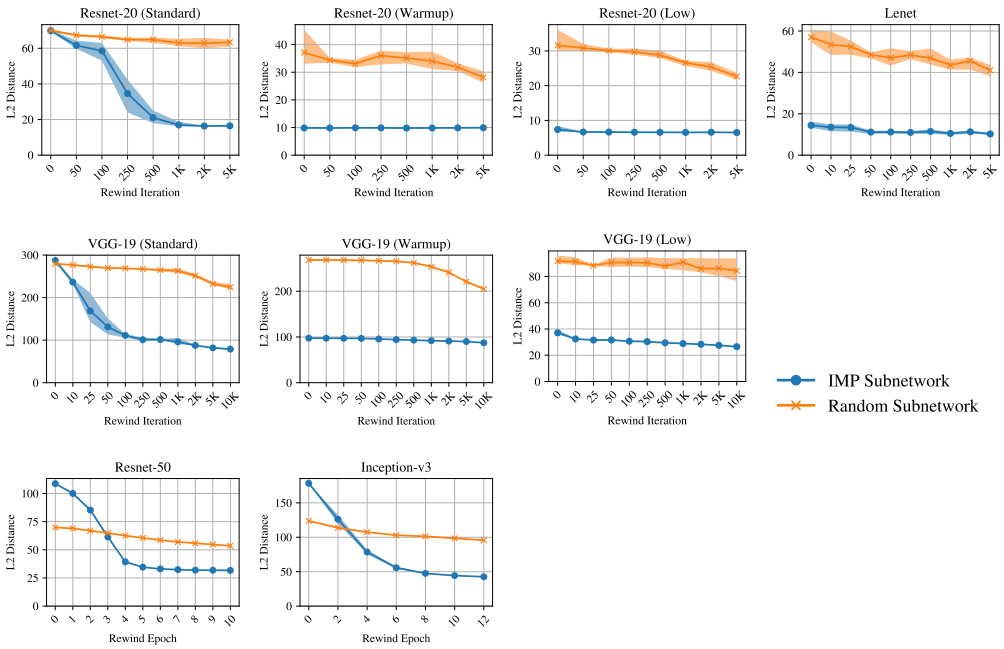
This subsection reproduces the interpolation summary plots from Figures 3 and 5 in a single location.



A.3 L_2 Distance Plots

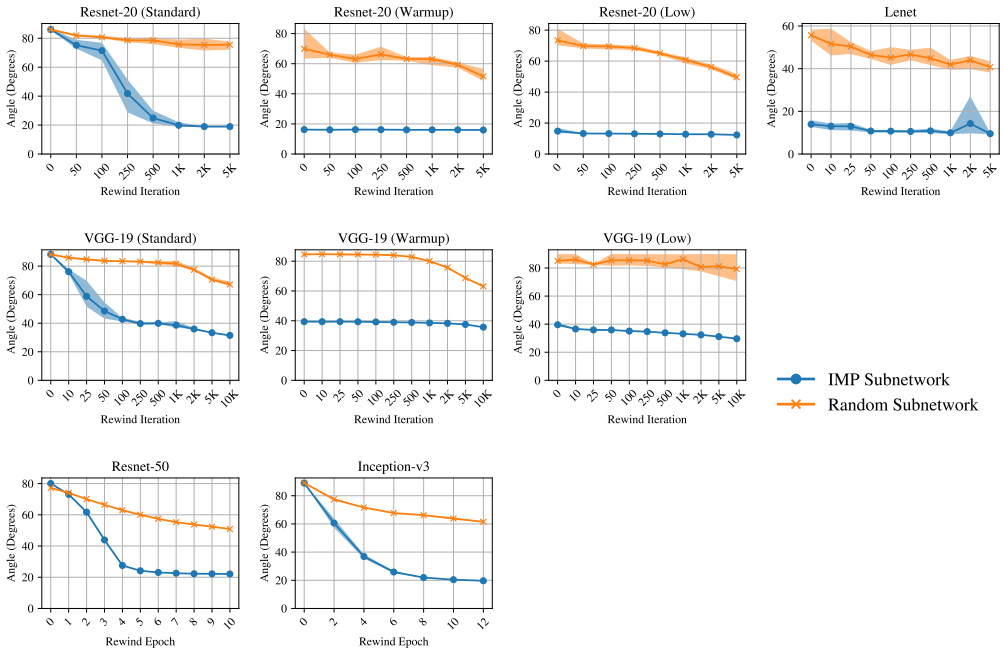
This subsection plots the L_2 distance between the final trained weights of copies of the subnetworks trained on different data orders. The trends with L_2 distance match those we observe for test error and stability: L_2 distance drops as test error decreases, saturating at the same time the subnetwork becomes stable. This data is further evidence that the optimization behavior of the stable subnet-

works is robust to data order noise: not only do the duplicates reach the same optimum, but they are also closer together in space.



A.4 Angle Distance Plots

This subsection plots the angle between the final trained weights of copies of the subnetworks trained on different data orders. As in Appendix A.3, the trends in these plots match those we see with test error and stability.



B COMPARING THE FULL NETWORK AND SUBNETWORK TRAINED WITH THE SAME DATA ORDER

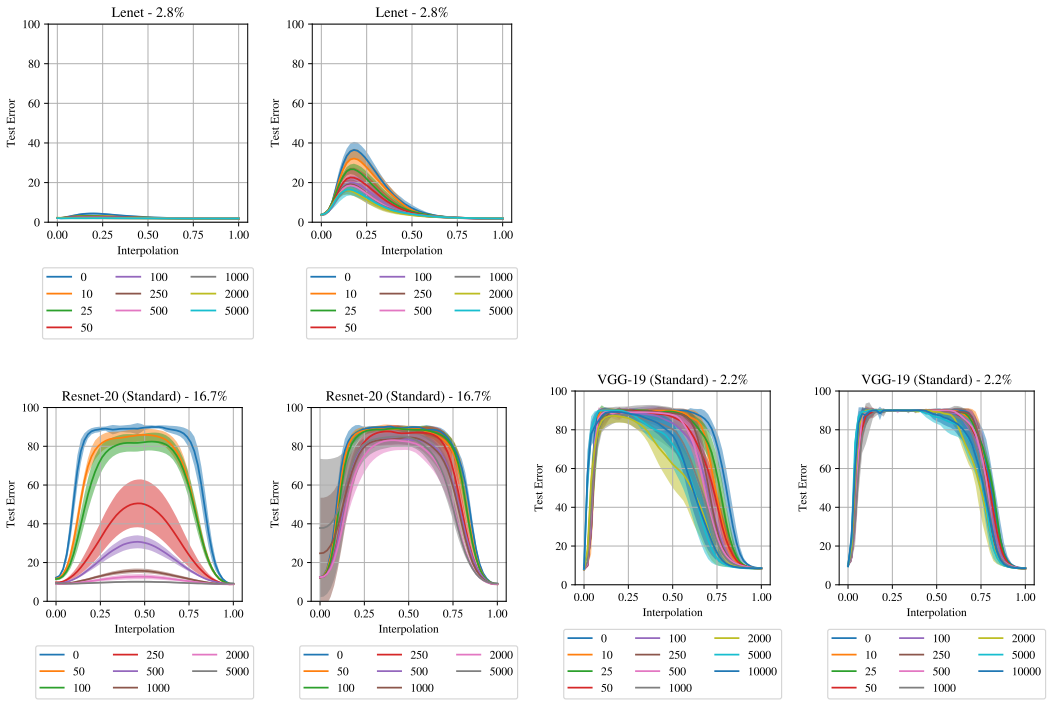
This appendix considers the relationship between the trained weights of the full network and subnetwork when they are trained using the same data order. One hypothesis for the relationship between accuracy and subnetwork stability is that the subnetworks are finding the same optimum as the full network. This appendix shows that this does not appear to be the case, at least under linear interpolation. Linearly interpolating between the full network and IMP subnetworks (Appendix B.1 for full data and Appendix B.2 for summary plots) traverses a region of increased test error, even when the subnetworks are stable and reach the maximum improvement in accuracy. However, as IMP subnetwork accuracy improves with rewinding, the subnetworks are closer in L_2 distance and angle to the trained weights of the full network, suggesting that the subnetworks may more closely follow the optimization trajectory of the full network, even if they do not end up in an optimum that is linearly connected.

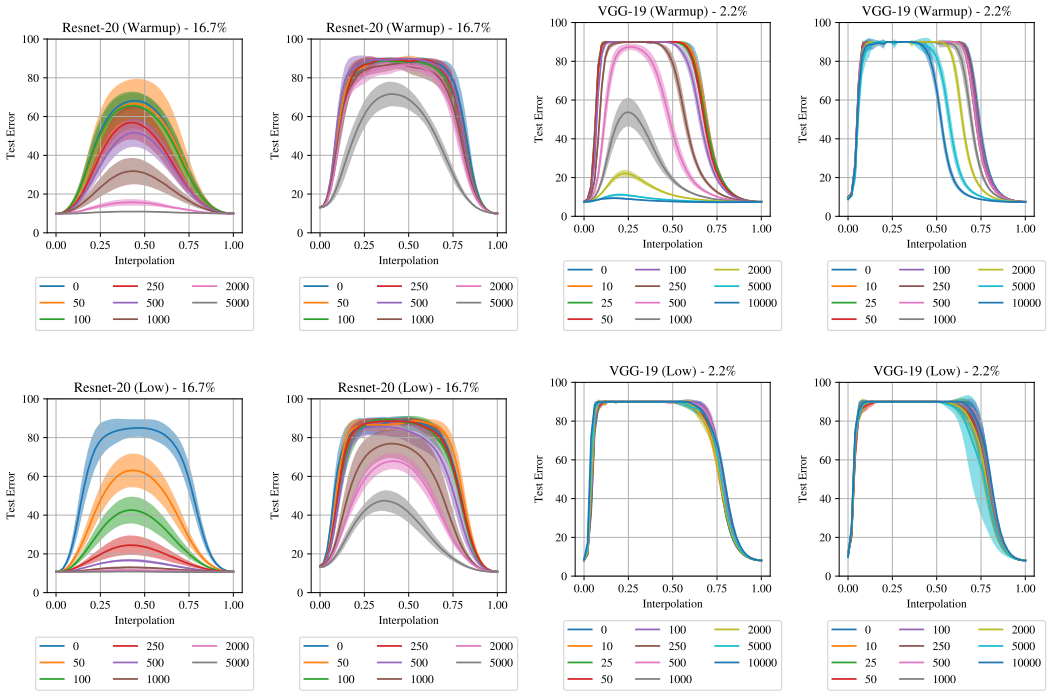
Other notes:

- We did not perform these interpolation experiments for the ImageNet networks.
- To compute L_2 distance and angle between the full network and subnetwork, we only consider weights that are not pruned. In other words, we compare trained subnetwork $\mathcal{A}^{t\to T}(m \odot W_t, u)$ to the trained full network with weights masked out $m \odot \mathcal{A}^{t\to T}(W_t, u)$.
- All lines are the average and standard deviation of three experiments (three initializations).

B.1 FULL INTERPOLATION PLOTS

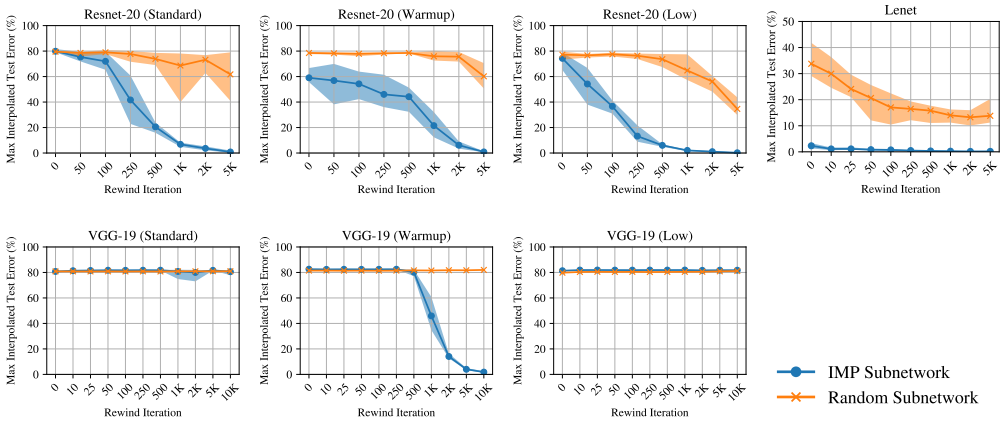
This subsection presents the test error when interpolating between the trained weights of the full network and subnetwork when trained on different data orders. For Resnet-20, the maximum increase in test error is generally smaller for IMP subnetworks than for random subnetworks, but we do not find a relationship between this increase and the test error of the subnetworks. For VGG-19, the maximum increase in test error is no smaller for IMP subnetworks than for random subnetworks except at the latest rewinding iterations under warmup. This data does not support the hypothesis that the pruned subnetworks find the same minimum as the full subnetwork, at least as far as can be determined by linear interpolation.





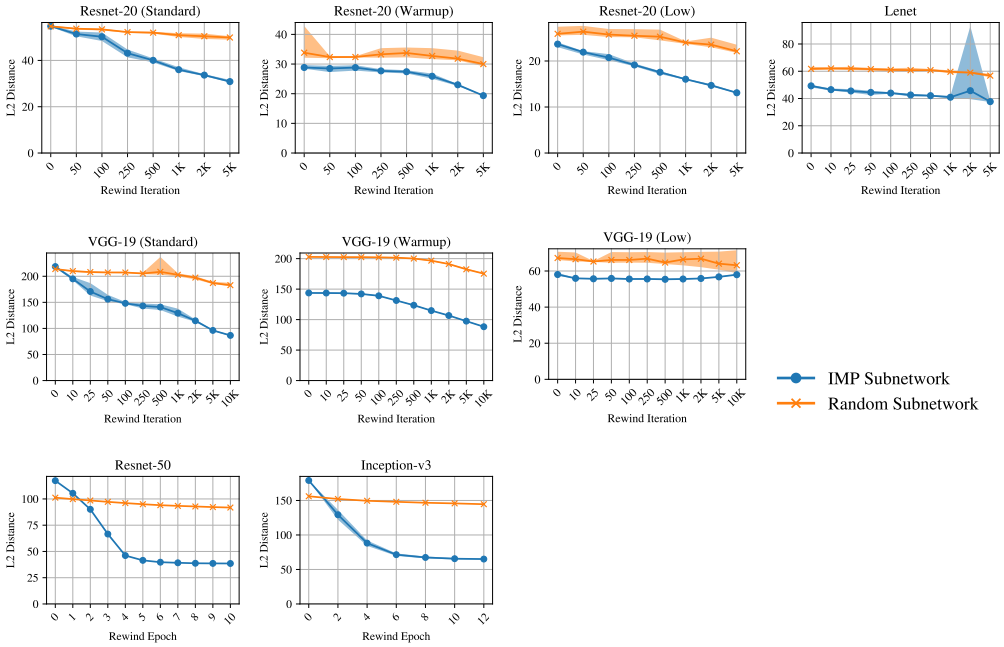
B.2 Interpolation Summary Plots

This subsection presents the summaries of the plots in Appendix B.1.



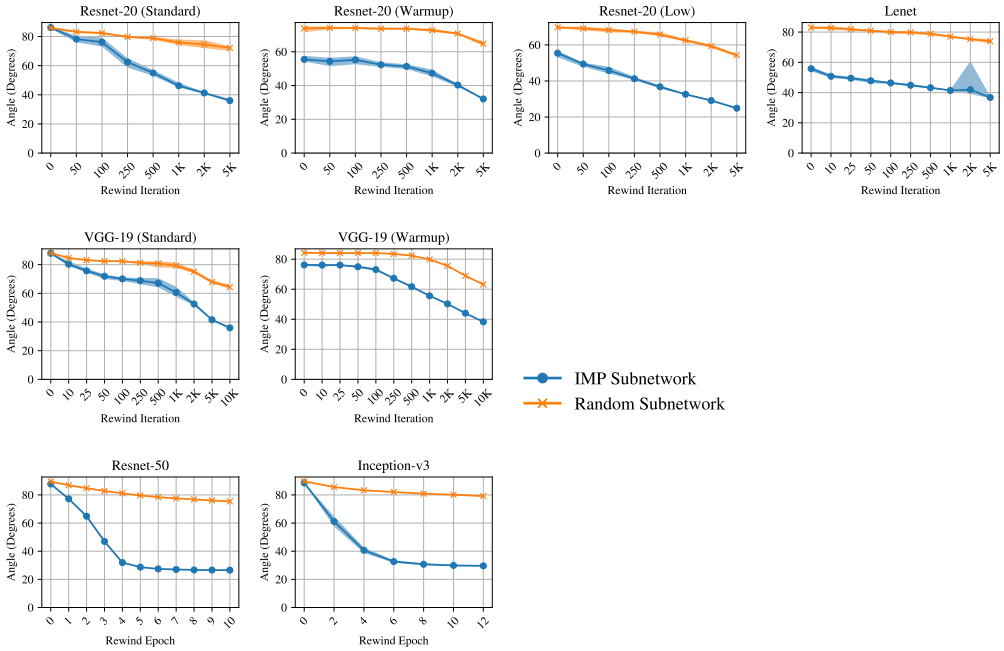
B.3 L_2 Distance Plots

This subsection plots the L_2 distance between the final trained weights of the full network and subnetwork trained on the same data order. To compute this distance, we ignore any weights in the full network that were pruned in the subnetwork. The trends with L_2 distance generally match the trends we see for test error and stability. In cases where we find matching subnetworks at all rewinding iterations, the IMP subnetwork weights are consistently closer to those of the full network than are the weights of the randomly pruned subnetworks. In cases where we need to rewind later in training to find matching subnetworks (Resnet-20, VGG-19, Resnet-50, and Inception-v3), there is no difference in L_2 distance between the IMP subnetworks and random subnetworks; as IMP subnetwork instability decreases and test accuracy improves, the L_2 distance also decreases, saturating at about the same iteration.



B.4 Angle Distance Plots

This subsection plots the angle between the final trained weights of copies of the subnetworks trained on different data orders. As in Appendix B.3, the trends in these plots match those we see with test error and stability.



C Comparing Full Networks Trained with Different Data Orders

In this Appendix, we consider interpolating between copies of the full network trained with different data orders. We use the same structure as the rewinding experiments: we train the full network to iteration k, make two copies of the network, and train those copies to completion with different data

orders. Appendix C.1 plots the errors at 50 points along the line between the trained weights of pairs of networks. Appendix C.2 summarizes the maximum increase in error across all rewinding iterations. Each line is the average and standard deviation of 18 samples (three initializations and three data orders per initialization). We do not include results for the ImageNet networks.

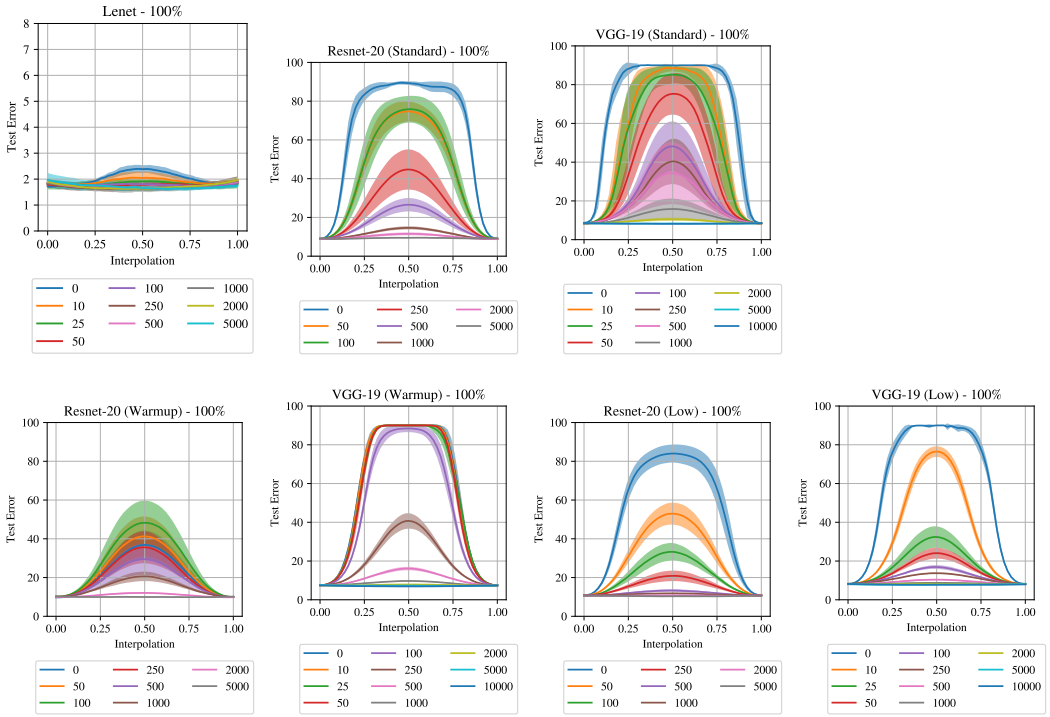
With the exception of Lenet, all networks are unstable at the earliest rewinding iterations. Beyond a certain rewinding iteration, all networks become stable. This trend is in contrast to what we observe with IMP subnetworks (blue lines), which—in many cases—are stable even at rewinding iteration 0. The rewinding iteration at which full network instability decreases varies from network to network. For Resnet-20 standard and VGG-19 standard, it occurs slightly later than when the IMP subnetworks become stable. Warmup causes stability to improve later, and a lower learning causes instability to improve slightly earlier.

Our first observation is that full networks do eventually become stable in all cases, and the point at which they do so occurs relatively early in training. That stability emerges early in training is not merely a property of extremely sparse IMP subnetworks—it is a property of SGD on these particular networks in general.

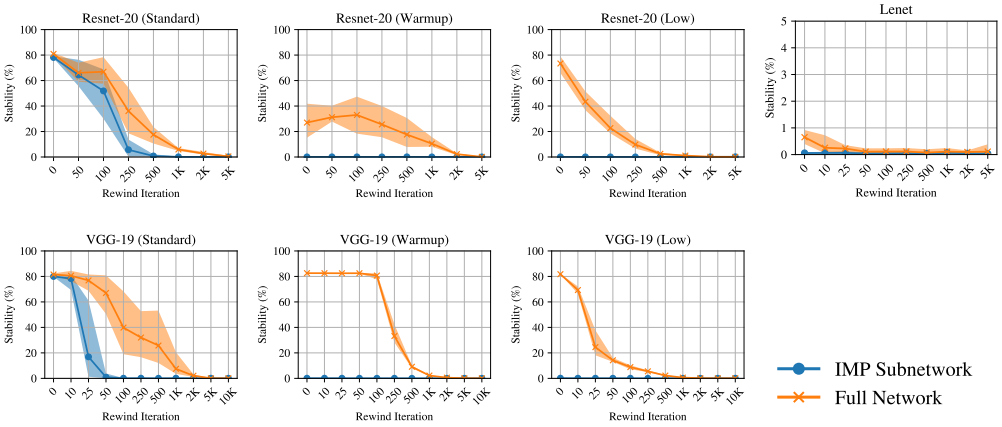
However, the stability behavior of the full networks differs from that of the IMP subnetworks. They become stable at different points, and—in some cases—the IMP subnetworks are never unstable while the full networks are at early rewinding points. On Resnet-20 standard and VGG-19 standard, the stability improvements for IMP subnetworks and the full networks occur in a similar pattern, albeit slightly delayed for the full networks, potentially indicating that there may be some relationship between the stability behavior of the subnetwork and the full network.

One possible explanation for this data is that the overparameterized full networks have more possible directions in the optimization landscape that they could pursue, meaning noise early in training may have a higher probability of changing the optimization trajectory without affecting the eventual performance of the trained network. In contrast, extremely sparse subnetworks have many fewer possible directions, meaning their optimization trajectories may become more robust to noise earlier in training.

C.1 Full Interpolation Plots



C.2 Interpolation Summary Plots



D Sparsity Levels During Training for Pruning Methods and IMP

Figure 7 shows the sparsity levels achieved during training by gradual pruning (Gale et al., 2019), training a pruned subnetwork from scratch with a new initialization (Liu et al., 2019), and the sparsity levels found by IMP. We do not propose IMP as an efficient pruning technique, as it requires training the full network many times to achieve these sparsity levels. However, the sparse subnetworks found early in training by IMP are proof that there is an unexploited opportunity to prune earlier in training than current methods do.

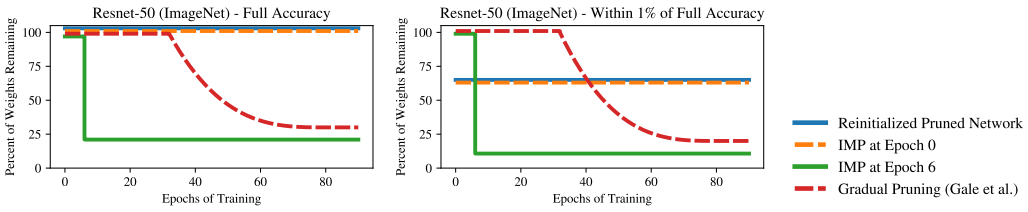


Figure 7: Sparsity levels during training (Resnet-50 on ImageNet) using a reinitialized pruned network (Liu et al. (2019) with our experiments in Section 5) and gradual pruning (numbers from Gale et al. (2019)) alongside the sparsity of subnetworks found retroactively at epochs 0 and 6 by IMP. We selected the sparsest network available such that top-1 accuracy met the specified constraint.