Stabilizing the Lottery Ticket Hypothesis

Jonathan Frankle
MIT CSAIL

Karolina Dziugaite
University of Cambridge
Element AI

Daniel M. Roy
University of Toronto
Vector Institute

Michael Carbin
MIT CSAIL

Abstract

Pruning is a well-established technique for removing unnecessary structure from neural networks after training to improve the performance of inference. Several recent results have explored the possibility of pruning at initialization time to provide similar benefits during training. In particular, the *lottery ticket hypothesis* conjectures that typical neural networks contain small subnetworks that can train to similar accuracy in a commensurate number of steps. The evidence for this claim is that a procedure based on iterative magnitude pruning (IMP) reliably finds such subnetworks retroactively on small vision tasks. However, IMP fails on deeper networks, and proposed methods to prune before training or train pruned networks encounter similar scaling limitations.

In this paper, we argue that these efforts have struggled on deeper networks because they have focused on pruning precisely at initialization. We modify IMP to search for subnetworks that could have been obtained by pruning *early* in training (0.1% to 7% through) rather than at iteration 0. With this change, it finds small subnetworks of deeper networks (e.g., 80% sparsity on Resnet-50) that can complete the training process to match the accuracy of the original network on more challenging tasks (e.g., ImageNet). In situations where IMP fails at iteration 0, the accuracy benefits of delaying pruning accrue rapidly over the earliest iterations of training. To explain these behaviors, we study subnetwork *stability*, finding that—as accuracy improves in this fashion—IMP subnetworks train to parameters closer to those of the full network and do so with improved consistency in the face of gradient noise. These results offer new insights into the opportunity to prune large-scale networks early in training and the behaviors underlying the lottery ticket hypothesis.

1 Introduction

For decades, *pruning* (LeCun et al., 1990; Han et al., 2015) unnecessary structure from neural networks has been a popular way to improve the storage and computational costs of inference, which can often be reduced by an order of magnitude without harm to accuracy. Pruning is typically a post-processing step after training; until recently, it was believed that the pruned architectures could not themselves be trained from the start (Han et al., 2015; Li et al., 2016). New results challenge this wisdom, raising the prospect of reducing the cost of training by pruning beforehand. Liu et al. (2019) demonstrate that, at moderate levels of sparsity, pruning produces networks that can be reinitialized and trained to equal accuracy; Lee et al. (2019) propose an efficient method for finding such reinitializable subnetworks before training (SNIP).

Frankle and Carbin (2019) characterize the opportunity for pruning at initialization. For shallow vision networks, they observe that—at levels of sparsity that are more extreme than Liu et al. (2019) and Lee et al. (2019)—pruned networks can successfully train from scratch so long as each unpruned connection is *reset* back to its initial value from before training.[1] This procedure, which we term *iterative magnitude pruning* (IMP; Algorithm 1 with $k = 0$), produces a subnetwork of the original, untrained network; when it matches the accuracy of the original network, it is called a *winning ticket*. Based on these results, Frankle and Carbin propose the *lottery ticket hypothesis*: dense neural networks contain sparse subnetworks capable of training to commensurate accuracy at similar speed.

[1] Appendix A compares Liu et al. (2019), Lee et al. (2019), and Frankle and Carbin (2019).

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

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: Train the network for $T - k$ further iterations, producing network $f(x; m \odot W_t)$.
4: Prune the remaining entries with the lowest magnitudes from W_T. That is, let $m[i] = 0$ if $W_T[i]$ is pruned.
5: If satisfied, the resulting network is $f(x; m \odot W_T)$.
6: Otherwise, reset W to W_k and repeat steps 3-5 iteratively, gradually removing more of the network.

Despite the enormous potential of pruning before training, none of this work scales beyond small vision benchmarks. Lee et al. provides results only for Tiny ImageNet, a restricted version of ImageNet with 200 classes. Liu et al. examine Resnet-50 on ImageNet, but accuracy declines when only 30% of parameters are pruned. Liu et al., Gale et al. (2019), and Frankle and Carbin themselves show that IMP fails on deeper networks. To find winning tickets on deeper networks for CIFAR10, Frankle and Carbin make bespoke changes to each network's learning schedule. In this paper, we argue that such efforts to train pruned networks or prune before training have struggled on deeper networks because they have focused on doing so precisely at initialization.

In comparison, other techniques gradually prune networks throughout training to competitive levels of sparsity without compromising accuracy (Zhu and Gupta, 2017; Gale et al., 2019; Lym et al., 2019; Narang et al., 2017; Louizos et al., 2018; Narang et al., 2017). However, these approaches must maintain much of the network for a large portion of training or do not scale beyond toy benchmarks.

Rewinding. In this paper, we demonstrate that there exist subnetworks of deeper networks (i.e., Resnet-50, Squeezenet, Inception-v3) at early points in training (0.1% to 7% through) that are 50% to 99% smaller and that can complete the training process to match the original network's accuracy. We show this by modifying IMP to *rewind* pruned subnetwork weights to their former values at iteration k rather than *resetting* them to iteration 0. For networks where IMP cannot find a winning ticket, the accuracy benefits of this delay in pruning accrue rapidly over the earliest iterations of training. For example, IMP finds 80% sparse subnetworks of Resnet-50 at epoch 6 (out of 90) with no loss in accuracy on ImageNet. To the best of our knowledge, our work is the first to show that it is possible to prune (1) so early in training (2) to such extreme levels of sparsity (3) on such large-scale tasks.

Stability. To explain why IMP fails when resetting to iteration 0 and improves rapidly when rewinding later, we introduce subnetwork *stability*: the distance between two trained copies of the same subnetwork subjected to different noise. In particular, we focus on the noise introduced by pruning (comparing the trained weights of the full network and subnetwork) and data order (comparing the weights of two subnetworks trained with different data orders). In cases where IMP fails to find a winning ticket when resetting to iteration 0, both forms of stability improve rapidly as pruning is delayed during the early part of training, mirroring the rise in accuracy. Stability to pruning captures the extent to which the subnetwork arrived at the same destination as the original network in the optimization landscape. We hypothesize that improved stability to pruning means that a subnetwork comes closer to the original optimum and thereby accuracy; improvements in stability to data order mean the subnetwork can do so consistently in spite of the noise intrinsic to SGD.

Finally, we revise the lottery ticket hypothesis to consider rewinding in accordance with these results:

The Lottery Ticket Hypothesis with Rewinding. *Consider a dense, randomly-initialized neural network $f(x; W_0)$ that trains to accuracy a^* in T^* iterations. Let W_t be the weights at iteration t of training. There exist an iteration $k \ll T^*$ and fixed pruning mask $m \in \{0, 1\}^{|W_0|}$ (where $||m||_1 \ll |W_0|$) such that subnetwork $m \odot W_k$ trains to accuracy $a \geq a^*$ in $T \leq T^* - k$ iterations.*

Based on this new understanding of the lottery ticket hypothesis provided by our rewinding and stability experiments, we conclude that there are unexploited opportunities to prune large-scale networks early in training while maintaining the accuracy of the eventual trained networks.

2 Stability at Initialization

On deeper networks for CIFAR10, Iterative Magnitude Pruning (IMP; Algorithm 1 at $k = 0$) fails to yield winning tickets. The solid blue line in Figure 1 shows that IMP on VGG-19 (left) and Resnet-18 (right) produces no winning tickets; the original initialization is inconsequential, and the networks

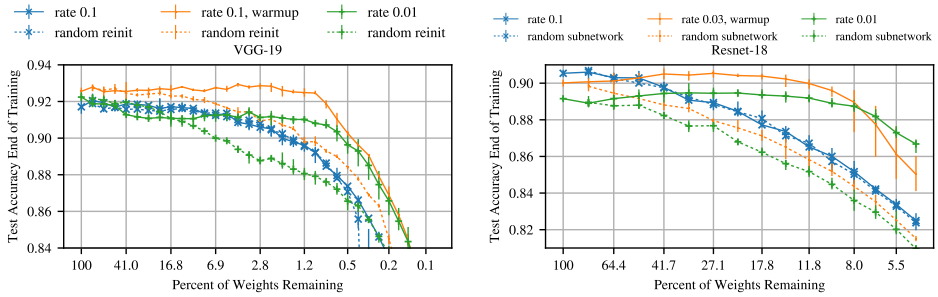

Figure 1: On deeper networks for CIFAR10, IMP fails to find winning tickets unless the learning rate schedule is altered.

Network	Dataset	Params	Iters	Batch	Accuracy	Rate	Schedule	Warmup	Winning?
Lenet	MNIST	266K	50K	60	98.0 ± 0.2%	adam 12e-4	constant	0	Y
Resnet-18 (standard)	CIFAR10	274K	30K	128	90.5 ± 0.0%	mom. 0.1	10x drop at 56K, 84K	0	N
Resnet-18 (low)					89.1 ± 0.1%	mom. 0.01		0	Y
Resnet-18 (warmup)					89.6 ± 0.1%	mom. 0.03		20K	Y
VGG-19 (standard)	CIFAR10	20.0M	112K	64	91.5 ± 0.2%	mom. 0.1	10x drop at 20K, 25K	0	N
VGG-19 (low)					92.0 ± 0.1%	mom. 0.01		0	N
VGG-19 (warmup)					92.3 ± 0.1%	mom. 0.1		10K	Y

Figure 2: Networks for MNIST and CIFAR10. Accuracies averaged across three trials.

can be reinitialized (dashed blue) without altering accuracy. Frankle and Carbin manage to find winning tickets in these networks by altering the learning schedule, lowering the learning rate (green) or adding warmup (orange). However, they offer no principled justification for these choices, which are brittle and often alter the behavior of the unpruned networks.

To understand why IMP succeeds or fails to find winning tickets, we examine their *stability*. We measure two forms of stability: 1) *stability to pruning:* the distance between the weights of a subnetwork trained in isolation and the weights of the same subnetwork when trained within the larger network and 2) *stability to data order:* the distance between the weights of two copies of a subnetwork trained with different data orders. Stability to pruning captures a subnetwork's ability to train in isolation and still reach the same destination as the larger network. Stability to data order captures a subnetwork's intrinsic ability to consistently reach the same destination despite the gradient noise of SGD. In this section, we demonstrate that, when IMP returns a subnetwork that qualifies as a winning ticket, it is dramatically more stable by both measures than a randomly-pruned network of the same size.

Formal definitions. A *subnetwork* is a tuple (W, m) of weights $W : \mathbb{R}^D$ and a fixed pruning mask $m : \{0, 1\}^D$. Notation $m \odot W$ denotes an element-wise product of a mask with weights. A stochastic training *algorithm* $\mathcal{A}^t : \mathbb{R}^D \times U \to \mathbb{R}^D$ maps initial weights W and data order randomness $u \sim U$ to weights W_t at iteration $t \in \{1, .., T\}$. The distance d between trained weights W_t and W'_t is the L_2 distance between the masked, trained parameters: $||m \odot W_t - m \odot W'_t||_2$. Throughout the paper, Appendix B, follows the same analysis for the angle between the masked, trained parameters.

The stability to pruning of a subnetwork (W_t, m) with respect to a noise distribution U under a distance $d(\cdot, \cdot)$ is the expected distance between masked weights at the end of training: $d(\mathcal{A}^{T-t}(W_t, u), \mathcal{A}^{T-t}(m \odot W_t, u))$ for $u \sim U$.

The stability to data order of a subnetwork (W_t, m) with respect to a noise distribution U under a distance $d(\cdot, \cdot)$ is the expected distance between masked weights at the end of training: $d(\mathcal{A}^{T-t}(m \odot W_t, u), \mathcal{A}^{T-t}(m \odot W_t, u'))$ for $u, u' \sim U$.

Methodology. We measure both forms of stability for Lenet for MNIST and Resnet-18 and VGG-19 for CIFAR10 as studied by Frankle and Carbin and described in Figure 2. We do so for networks produced by IMP as well as randomly-pruned networks in which we generate a random mask m of a given size with the same layer-wise proportions. These networks are trained for a fixed number of epochs. During each epoch, all training examples are randomly shuffled, randomly augmented,

Network	Sparsity	Data Order Stability (Distance)			Pruning Stability (Distance)			Accuracy	
		IMP	Random	Comp	IMP	Random	Comp	IMP	Random
Lenet	10.7%	20.7 ± 0.6	58.6 ± 4.0	2.8x	48.1 ± 0.6	75.7 ± 0.5	1.6x	98.3 ± 0.1	97.5 ± 0.3
Resnet-18 (standard)	16.7%	66.4 ± 0.7	66.7 ± 1.1	1.0x	54.4 ± 0.2	53.4 ± 0.4	1.0x	87.7 ± 0.4	87.7 ± 0.5
Resnet-18 (low)		7.1 ± 1.2	28.9 ± 3.2	4.1x	19.8 ± 1.4	26.6 ± 0.1	1.3x	89.1 ± 0.4	86.1 ± 0.6
Resnet-18 (warmup)		9.5 ± 0.1	37.4 ± 1.9	3.9x	24.8 ± 0.9	34.6 ± 0.2	1.4x	90.3 ± 0.4	86.8 ± 0.5
VGG-19 (standard)	2.2%	285 ± 3	245 ± 34	0.8x	216 ± 1	212 ± 1	1.0x	90.0 ± 0.3	90.2 ± 0.5
VGG-19 (low)		36.8 ± 2.6	90.3 ± 0.7	2.5x	44.0 ± 0.3	66.1 ± 0.4	1.5x	91.0 ± 0.3	88.0 ± 0.3
VGG-19 (warmup)		97 ± 0.6	267 ± 2	2.7x	138 ± 1	201 ± 1	1.5x	92.4 ± 0.2	90.3 ± 0.3

Figure 3: The average data order stability of subnetworks obtained by IMP and by randomly pruning. Errors are the minimum or maximum across 18 samples.

and separated into minibatches without replacement; network parameters are updated based on each minibatch in sequence until the training data is exhausted, after which the next epoch begins.

Results. Figure 3 displays the stability for IMP subnetworks at a representative level of sparsity. IMP finds winning tickets for Lenet, but cannot do so for Resnet-18 or VGG-19 without warmup. Whenever it does so, the winning ticket is far more stable than a randomly-sampled subnetwork. For example, Lenet winning tickets are 2.8x (data order) and 1.6x (pruning) closer in L_2 distance than random subnetworks. Conversely, IMP cannot find a winning ticket within VGG-19 (standard) and Resnet-18 (standard), and the subnetworks it returns are no more stable than random subnetworks.

Discussion. These results suggest a connection between winning tickets and stability. Winning tickets are far more stable than random subnetworks. Likewise, when IMP subnetworks are no more accurate than random subnetworks, they are correspondingly no more stable. Interestingly, Frankle and Carbin's techniques that enable IMP to find winning tickets also benefit stability.

3 Stability with Rewinding

In cases where the IMP fails to find a winning ticket, reducing the learning rate early in training via warmup makes it possible for the procedure to succeed, simultaneously improving the stability of the resulting subnetwork. The efficacy of warmup suggests that a high learning rate is not necessarily detrimental later in training—only in the initial iterations. One hypothesis to explain these results is that—for the less overparameterized regime of training a subnetwork—optimization is particularly sensitive to noise in the early phase of training. This sensitivity can be mitigated by reducing the learning rate early on, minimizing the consequences of any individual misstep.

Under this hypothesis, we would expect these subnetworks to become more stable later in training, when they better tolerate a high learning rate. We explore this possibility by modifying IMP (Algorithm 1 with $k \neq 0$). After finding a subnetwork, *rewind* each connection back to its weight from an earlier iteration k rather than *resetting* it back to its initial value as Frankle and Carbin do.

Results. The upper two plots for each network in Figure 4 show the stability (measured in distance) of the IMP subnetwork (blue) and a random subnetwork (orange) across rewinding iterations. Across all networks, rewinding later causes gradual improvements in stability, supporting our hypothesis.

For Resnet-18 (standard) and VGG-19 (standard), in which no winning tickets can be found, IMP subnetworks are no more stable than randomly-pruned subnetworks when reset to iteration 0. However, as the rewinding iteration increases, the IMP subnetwork's stability dramatically improves when compared to its stability at iteration 0. Up to our point of analysis, this improvement is larger than that for random subnetworks. For IMP subnetworks, this change takes place rapidly—within the first 100 iterations (0.14 epochs) for VGG-19 and 500 iterations (1.4 epochs) for Resnet-18.

In cases where IMP finds a winning ticket, IMP subnetworks are already more stable than random subnetworks when resetting to iteration 0 (as noted in Section 2). This stability gap remains in place across all rewinding iterations that we consider, although it gradually shrinks toward the end of our range of analysis as the random subnetworks become somewhat more stable; by this point, the networks have already made substantial training progress or—in the case of Lenet—converged.

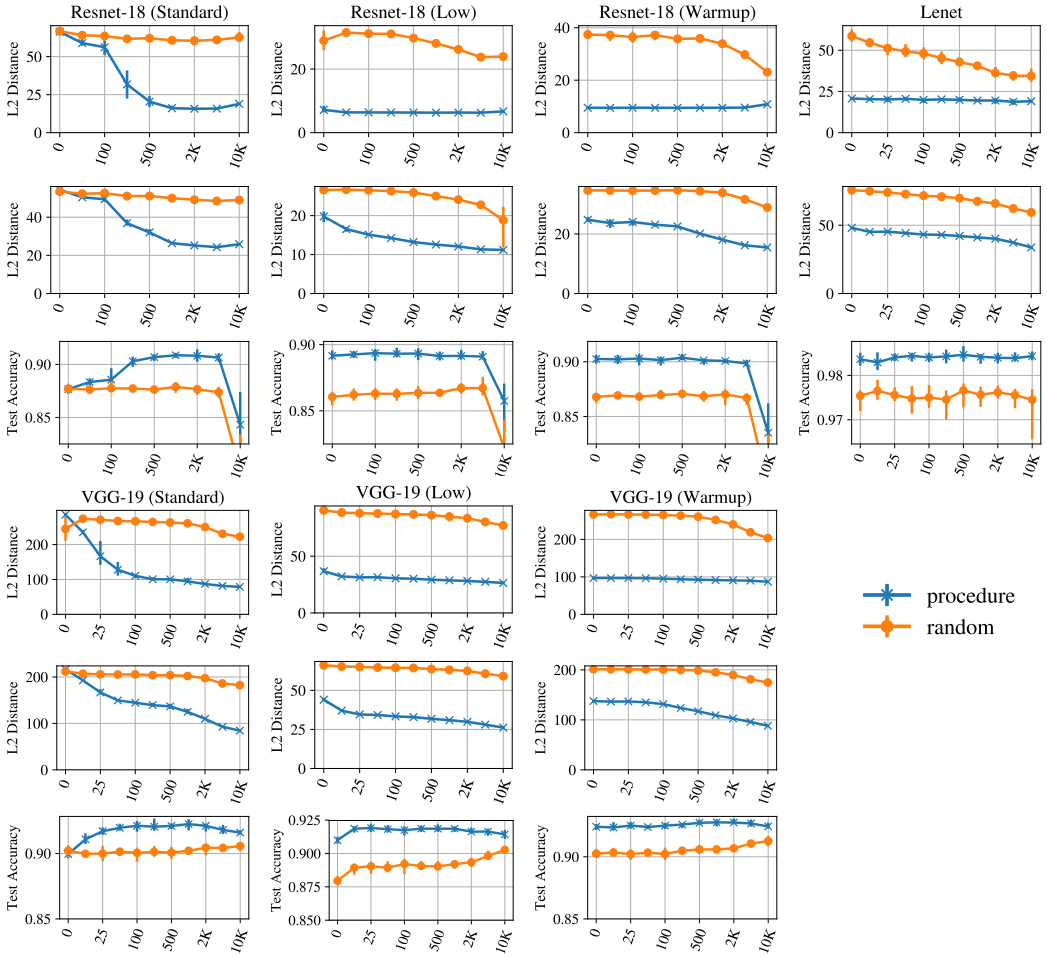

Figure 4: The effect of the rewinding iteration (x-axis) on data order (top row) and pruning (middle row) stability and accuracy (bottom row) for networks in Figure 2. Error bars are the minimum and maximum across 18 (data order) and three samples (stability and accuracy). These results were obtained by training more than 6500 networks on K80s and V100s.

Discussion. Section 2 shows that, when IMP identifies a winning ticket, it is more stable than a random subnetwork. Here, we find that IMP subnetworks generally achieve such a stability gap across all network configurations; however, in many cases, this occurs a small way into training.

4 Accuracy with Rewinding

Section 2 observes that winning tickets found by IMP exhibit both improved accuracy and stability over random subnetworks. The previous section finds that the stability of IMP subnetworks improves as a function of rewinding iteration, especially rapidly so for networks where winning tickets are not found. The bottom plots in Figure 4 show that accuracy improves in a similar manner. Although resetting to iteration 0 does not produce winning tickets for Resnet-18 and VGG-19, rewinding slightly later (iteration 500 for Resnet-18 and 100 for VGG-19) leads to IMP subnetworks that exceed the accuracy of the original network. At later rewinding iterations, the rate of stability improvements subsides for these subnetworks, corresponding to slowed or no improvement in accuracy.

Improved stability also results in improved accuracy for random subnetworks. However, the accuracy of the random subnetworks remains lower in all of our experiments. We believe that the significant stability gap explains this difference: we hypothesize that the stability of IMP subnetworks over their random counterparts results in better accuracy.

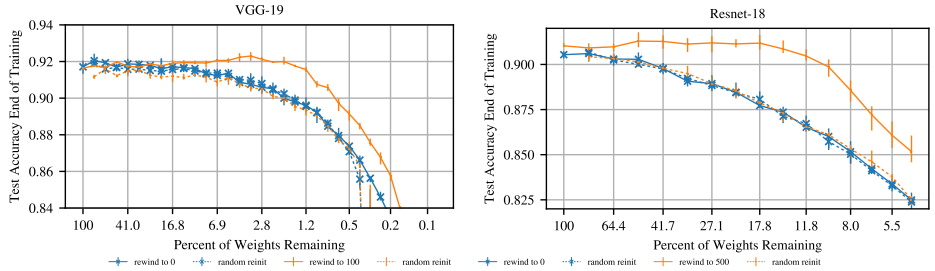

Figure 5: IMP subnetworks rewound to an iteration early in training outperform the original networks, while resetting to iteration 0 does not.

Network	Params	Eps.	Batch	Top-1 Accuracy	Learning Schedule	Sparsity
Resnet-50	25.5M	90	1024	76.1 $\pm$ 0.1%	rate 0.4; warmup 5 eps.; 10x drop at 30, 60, 80; momentum	70%
Inception-v3	27.1M	171	1024	78.1 $\pm$ 0.1%	rate 0.03 linearly decayed to 0.005; momentum	70%
Squeezenet	1.25M	150	1024	54.8 $\pm$ 0.6 %	rate 0.66 exponentially decayed to 6.6e-5; rmsprop	50%

Figure 6: Networks for experiments on ImageNet with TPUs. Accuracies averaged across five trials.

For the latest rewinding iterations that we consider, IMP subnetwork accuracy declines in many cases. According to our methodology, we train a subnetwork rewound to iteration k for $T^* - k$ iterations, where T^* is the iterations for which the original network was trained. At later rewind points, we believe that this does not permit enough training time for the subnetwork to recover from pruning.

Discussion. Figure 5 plots the accuracy of Resnet-18 and VGG-19 across all levels of sparsity, comparing the IMP subnetworks reset to iteration 0 with those rewound to iteration 100 (VGG-19) and iteration 500 (Resnet-18)—the iterations at which stability and accuracy improvements saturate. At all levels of sparsity, rewinding makes it possible to find trainable subnetworks early in training without any modifications to network hyperparameters (unlike the warmup or reduced learning rates required in Figure 1).

These findings reveal a new opportunity to improve the performance of training. The aspiration behind the lottery ticket hypothesis is to find small, trainable subnetworks before any training has occurred. Insofar as IMP reflects the extent of our knowledge about the existence of equally-capable subnetworks early in training, our findings suggest that—for deeper networks—the best opportunity to prune is a small number of iterations into training rather than at initialization. Doing so would exploit the rapid improvements in subnetwork stability and accuracy, resulting in subnetworks that can match the performance of the original network at far greater levels of sparsity.

5 Rewinding on Deep Networks for ImageNet

Rewinding made it possible to find sparse, trainable subnetworks of deeper networks for CIFAR10 without the need to alter the underlying network's hyperparameters. In this section, we demonstrate that this strategy serves the same purpose for deeper networks for ImageNet (Russakovsky et al., 2015). Although IMP with $k = 0$ does not produce winning tickets, rewinding 4.4%, 3.5%, and 6.6% into training yields subnetworks that are 70%, 70%, and 50% smaller than the Resnet-50 (He et al., 2016), Inception-v3 (Szegedy et al., 2016), and Squeezenet (Iandola et al., 2016) architectures, respectively, that can complete training without any drop in accuracy. We trained more than 600 networks using standard implementations for TPUs (Google, 2018) as described in Figure 6.

Figure 7 shows the effect of the rewinding iteration on the stability and accuracy at the levels of sparsity from Figure 6. In general, the trends from Section 4 remain in effect. When resetting weights to initialization, IMP subnetworks perform no better than random subnetworks in terms of both stability and accuracy. As the rewinding iteration increases, a gap in stability emerges. Accuracy improves alongside stability, reaching the accuracy of the original network at epoch 4 (out of 90) for Resnet-50, 6 (out of 171) for Inception-v3, and 10 (out of 150) for Squeezenet. In the case of Squeezenet, rewinding too early makes it impossible for the subnetworks to learn at all.

Figure 8 illustrates the effect of performing IMP with rewinding across all levels of sparsity. The blue line shows the result of *one-shot pruning* (pruning all at once after training) at a rewinding

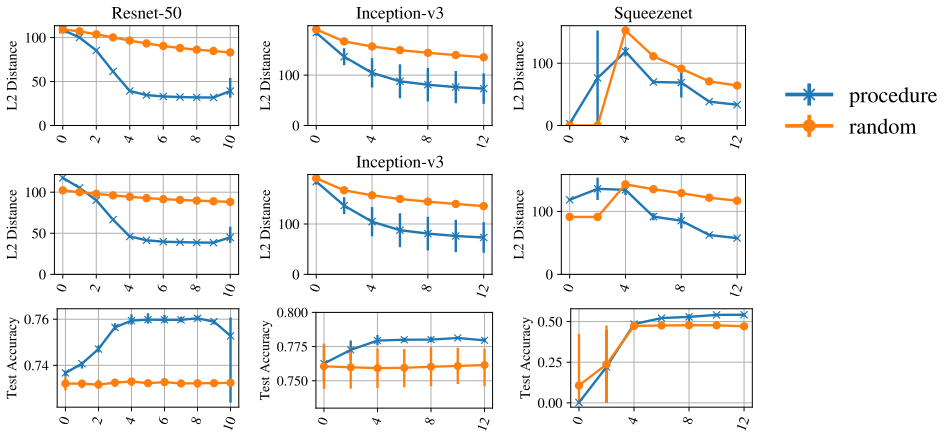

Figure 7: The effect of the rewinding epoch (x-axis) on data order stability and accuracy.

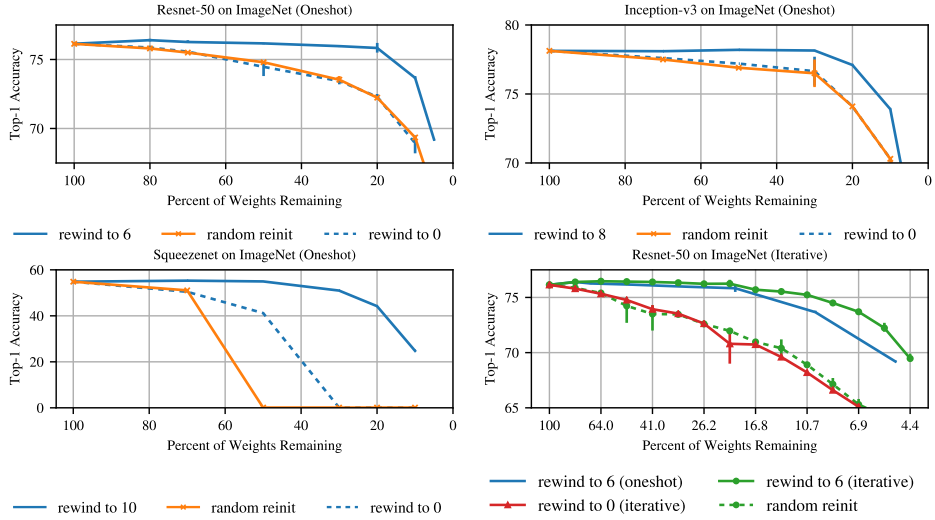

Figure 8: Rewinding to iterations early in training produces subnetworks that outperform the original networks, even when resetting to iteration 0 does not.

iteration just after the aforementioned thresholds. Resnet-50, Squeezenet, and Inception match the accuracy of the original network when 70%, 50%, and 70% of weights are pruned. At lower levels of sparsity, these subnetworks slightly outperform the original network. The weights obtained by rewinding are essential: when randomly reinitialized (dashed blue line) or reset to iteration 0 (orange line), subnetworks lose accuracy when pruned by any amount.

The bottom right plot explores the effect of iterative pruning (training, pruning 20% of weights at a time, rewinding, and repeating) on Resnet-50. Rewinding to epoch 6 (the green line) makes it possible to find subnetworks that train to match the original accuracy when just 20% of weights remain. Randomly reinitializing (dashed green line) or resetting to iteration 0 (red line) perform equally poorly, only losing accuracy as they are pruned.

Discussion. On these deeper networks for a more challenging task, IMP finds no evidence in support of Frankle and Carbin's hypothesis that equally-capable subnetworks exist at initialization. However, rewinding to within a few epochs of the start of training makes it possible to find subnetworks with these properties. Stability continues to offer insight into the value of rewinding: as IMP subnetworks become more stable and a stability gap emerges, they reach higher accuracy. The central conceit of the lottery ticket hypothesis—that we can prune early in training—continues to apply in this setting; however, the most productive moment at which to do so is later than strictly at initialization.

6 Limitations

Like Frankle and Carbin's work, we focus only on image classification. While we extend their work to include ImageNet, the revised IMP must still train a network one or more times to identify a subnetwork; we do not propose an efficient way to find these subnetworks at the rewinding iteration. Our core pruning technique is still unstructured, magnitude pruning (among many other pruning techniques, e.g., Hu et al. (2016); Srinivas and Babu (2015); Dong et al. (2017); Li et al. (2016); Luo et al. (2017); He et al. (2017)). Unstructured pruning does not necessarily yield networks that execute more quickly with commodity hardware or libraries; we aim to convey insight on neural network behavior rather than suggest immediate opportunities to improve performance.

7 Discussion

Stability. Stability to pruning measures a subnetwork's ability to train in isolation to final weights that are close to the values they would have reached had they been trained as part of the original network. If we consider the trained weights of the original network to be an ideal destination for optimization, then this metric captures a subnetwork's ability to approach this point with fewer parameters. We conjecture that, by arriving at a similar destination, these subnetworks also reach similar accuracy, potentially explaining why increased stability to pruning corresponds to increased accuracy. Frankle and Carbin's winning tickets are substantially more stable to pruning than random subnetworks, shedding light on a possible mechanism behind the lottery ticket hypothesis. As a follow-up, one might explore whether subnetworks that are more stable to pruning are also more likely to reach the same basin of attraction as the original network (Nagarajan and Kolter, 2019).

We find a similar relationship between stability to data order and accuracy. Stability to data order measures a subnetwork's ability to reach similar final weights in spite of training with different mini-batch sequences—the gradient noise intrinsic to SGD. This metric is valuable because it provides a way to measure subnetwork stability without reference to the original network. We believe that stability to pruning and data order work hand-in-hand: subnetworks that are robust to data order are better able to reach destinations consistent with the original network in the face of SGD noise.

Pruning early. We aim to characterize the opportunity to prune early in training. Doing so could make it possible to reduce the cost of training networks by substantially reducing parameter-counts for most or all of training. This is an active area of research in its own right, including pruning before training (Lee et al., 2019), pruning throughout training (Zhu and Gupta, 2017; Gale et al., 2019; Lym et al., 2019; Molchanov et al., 2017; Louizos et al., 2018; Narang et al., 2017), and maintaining a sparse network with dynamic structure (Bellec et al., 2018; Mostafa and Wang, 2018; Mocanu et al., 2018). However, to the best of our knowledge, our work is the first to show that it is possible to prune (1) so early in training (2) to such extreme levels of sparsity (3) on such large-scale tasks.

Specifically, we find that, for many networks, there is an iteration early in training after which pruning can result in subnetworks with far higher accuracy than when pruning at initialization. Our results with IMP expand the range of known opportunities to prune early in training that—if exploited—could reduce the cost of training. With better techniques, we expect this range could be expanded even further because our results are restricted by IMP's limitations. Namely, it is possible that there are equally-capable subnetworks present at initialization, but IMP is unable to find them.

Stability offers a new perspective for developing new early pruning methods. One could exploit stability information for pruning, or develop new techniques to maintain stability under pruning. Under our hypothesized connection between stability and accuracy, such methods could make it possible to find accurate subnetworks early in training.

8 Conclusion

The lottery ticket hypothesis hints at future techniques that identify small, trainable subnetworks capable of matching the accuracy of the larger networks we typically train. To date, this and other related research have focused on compressing neural networks before training. In this work, we find that other moments early in the training process may present better opportunities for this class of techniques. In doing so, we shed new light on the lottery ticket hypothesis and its manifestation in deeper networks through the lens of stability.

References

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. 2018. Deep Rewiring: Training very sparse deep networks. *Proceedings of ICLR* (2018).

Xin Dong, Shangyu Chen, and Sinno Pan. 2017. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*. 4860–4874.

Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Int. Conf. Represent. Learn.* arXiv:1803.03635

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The State of Sparsity in Deep Neural Networks. *arXiv preprint arXiv:1902.09574* (2019).

Google. 2018. Networks for Imagenet on TPUs. (2018). `https://github.com/tensorflow/tpu/tree/master/models/`

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*. 1135–1143.

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

Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, Vol. 2. 6.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* (2016).

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

Yann LeCun, John S Denker, and Sara A Solla. 1990. Optimal brain damage. In *Advances in neural information processing systems*. 598–605.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2019. SNIP: Single-shot Network Pruning based on Connection Sensitivity. (2019).

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the Value of Network Pruning. In *International Conference on Learning Representations*. `https://openreview.net/forum?id=rJlnB3C5Ym`

Christos Louizos, Max Welling, and Diederik P Kingma. 2018. Learning Sparse Neural Networks through L_0 Regularization. *Proceedings of ICLR* (2018).

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. *arXiv preprint arXiv:1707.06342* (2017).

Sangkug Lym, Esha Choukse, Siavash Zangeneh, Wei Wen, Mattan Erez, and Sujay Shanghavi. 2019. PruneTrain: Gradual Structured Pruning from Scratch for Faster Neural Network Training. *arXiv preprint arXiv:1901.09290* (2019).

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications* 9, 1 (2018), 2383.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. 2017. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369* (2017).

Hesham Mostafa and Xin Wang. 2018. Dynamic parameter reallocation improves trainability of deep convolutional networks. (2018).

Vaishnavh Nagarajan and J. Zico Kolter. 2019. Uniform convergence may be unable to explain generalization in deep learning. *CoRR* abs/1902.04742 (2019). arXiv:1902.04742 `http://arxiv.org/abs/1902.04742`

Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. 2017. Exploring sparsity in recurrent neural networks. *Proceedings of ICLR* (2017).

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

Suraj Srinivas and R Venkatesh Babu. 2015. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149* (2015).

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878* (2017).

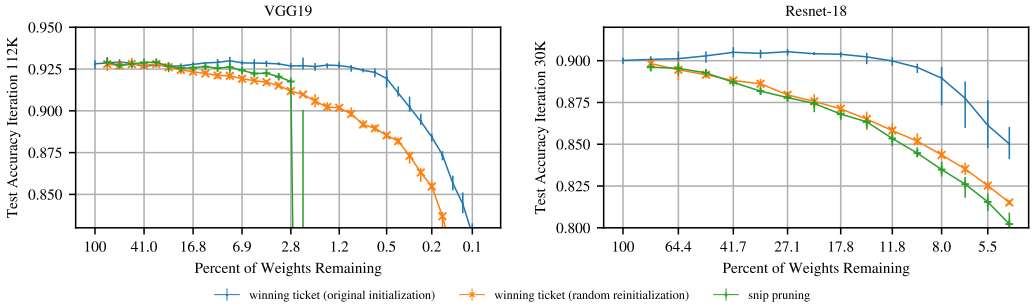

Figure 9: The accuracy achieved by VGG19 (left) and Resnet-18 (right) on CIFAR10 when pruned to the specified size using iterative pruning and SNIP. Networks are trained with warmup and the learning rate hyperparameters used by Frankle and Carbin.

A Comparison to "Rethinking the Value of Pruning" and "SNIP"

In this appendix, we compare the performance of subnetworks found by "The Lottery Ticket Hypothesis" (Frankle and Carbin, 2019), "Rethinking the Value of Pruning" (Liu et al., 2019), and "SNIP" (Lee et al., 2019). All three papers have different perspectives on the prospect of pruning early in training.

Frankle and Carbin argue that sparse, trainable networks exist at initialization time within the networks that we typically train. They find these networks using IMP (1 with $k = 0$) to find these subnetworks: they train the original network, prune it, and reset each surviving connection's weight back to its initial value from before training. They argue that the original initializations are essential for achieving this performance and that randomly reinitializing substantially degrades performance.

Liu et al. argue that the sparse networks that result from pruning can be trained from the start and that the original initializations do not matter. Instead, they find that the networks that result from pruning can be trained with a random initialization to the same performance as the original network.

Lee et al. propose a method for pruning early in training called SNIP. SNIP considers the *sensitivity* of the loss to each weight (based on one mini-batch of data) and removes those weights to which the loss is least sensitive. Sensitivity is measured by multiplying each weight w by a virtual parameter $c = 1$ and computing $\frac{\partial L}{\partial c}$. The authors find that SNIP can prune neural networks before they have been trained, and that these networks can be randomly reinitialized without harm to the eventual accuracy.

There are three points of contention between the papers:

1. Is the original initialization important? Frankle and Carbin argue that it is essential, but Liu et al. and Lee et al. discard it without any impact on their results.
2. At what level of sparsity are the authors measuring? For example, Liu et al. and Lee et al. consider VGG-19 when pruned by up to 95%, while Frankle and Carbin prune by upwards of 99.5%.
3. How efficient is the pruning method? Frankle and Carbin must train the same network a dozen or more times, while Liu et al. must train the network once and Lee et al. need to only look at a single mini-batch of data.

Figure 9 compares these three methods on VGG-19 (warmup) and Resnet-18 (warmup) from Figure 2. These particular hyperparameters were chosen because IMP does not find winning tickets without warmup, which would render this comparison less informative. The plots include the accuracy of randomly reinitialized winning tickets (orange), winning tickets with the original initialization (blue), and subnetworks found by SNIP (green). The results for VGG19 (left) support the findings of Liu et al. that pruned, randomly reinitialized networks can match the accuracy of the original network: VGG19 can do so when pruned by up to 80%. However, beyond this point, the accuracy of the randomly reinitialized networks declines steadily. In contrast, winning tickets with the original initialization match the accuracy of the original network when pruned by up to 99%. For Resnet-18 (right), which has 75x fewer parameters, the randomly reinitialized networks lose accuracy much sooner.

Network	Sparsity	Data Order Stability (Angle)			Pruning Stability (Angle)			Accuracy	
		IMP	Random	Comp	IMP	Random	Comp	IMP	Random
Lenet	10.7%	$22.7 \pm 0.5^\circ$	$49.2 \pm 3.0^\circ$	2.2x	$53.1 \pm 0.8^\circ$	$83.6 \pm 1.1^\circ$	1.6x	98.3 ± 0.1	97.5 ± 0.3
Resnet-18 (standard)	16.7%	$87.7 \pm 1.2^\circ$	$88.0 \pm 1.4^\circ$	1.0x	$87.7 \pm 1.2^\circ$	$88.0 \pm 1.4^\circ$	1.0x	87.7 ± 0.4	87.7 ± 0.5
Resnet-18 (low)		$16.1 \pm 2.7^\circ$	$75.8 \pm 1.2^\circ$	4.7x	$50.4 \pm 4.8^\circ$	$77.1 \pm 0.4^\circ$	1.5x	89.1 ± 0.4	86.1 ± 0.6
Resnet-18 (warmup)		$17.0 \pm 0.2^\circ$	$69.0 \pm 3.8^\circ$	4.7x	$49.9 \pm 2.3^\circ$	$79.0 \pm 0.5^\circ$	1.6x	90.3 ± 0.4	86.8 ± 0.5
VGG-19 (standard)	2.2%	$88.5 \pm 0.4^\circ$	$88.3 \pm 0.5^\circ$	1.0x	$87.8 \pm 0.2^\circ$	$88.1 \pm 0.2^\circ$	1.0x	90.0 ± 0.3	90.2 ± 0.5
VGG-19 (low)		$39.9 \pm 2.1^\circ$	$84.9 \pm 0.6^\circ$	2.1x	$54.0 \pm 0.4^\circ$	$79.0 \pm 0.2^\circ$	1.5x	91.0 ± 0.3	88.0 ± 0.3
VGG-19 (warmup)		$39.9 \pm 2.1^\circ$	$84.6 \pm 0.4^\circ$	2.1x	$71.7 \pm 0.3^\circ$	$84.8 \pm 0.2^\circ$	1.2x	92.4 ± 0.2	90.3 ± 0.3

Figure 10: The average stability (as measured in angle) of subnetworks obtained by IMP and by randomly pruning. Errors are the minimum or maximum across 18 samples (data order) and 3 samples (pruning and accuracy).

SNIP results in a promising improvement over random reinitialization on VGG19, however there is still a performance gap between SNIP and the winning tickets—an opportunity to further improve the performance of pruning before training.

We return to our original three questions: Up to a certain level of sparsity, the original initialization is not important. Subnetworks that are randomly reinitialized can still train to full accuracy. Beyond this point, however, the original initialization is necessary in order to maintain high accuracy. Liu et al. and Lee et al. operate in this first regime, where initialization is less important; Frankle and Carbin operate in the second regime. However, finding networks that learn effectively at these extreme levels of sparsity is very expensive: Frankle and Carbin must train the same network many times in order to do so. Lee et al. offer a promising direction for efficiently finding such subnetworks, taking a step toward realizing the opportunity described by Frankle and Carbin and this paper.

B Angle Measurements (Stability)

This appendix accompanies Figures 3 and 4, which measure the stability of the networks in Figure 2. The aforementioned figures measure stability only in terms of distance. This appendix includes the accompanying measurements of angle. Figure 10 includes angle data when resetting at iteration 0 to accompany Figure 3. Figure 11 includes the angle measurements of stability to data order (top) and pruning (middle) when networks are rewound to various iterations; it accompanies Figure 4. Figure 12 includes angle measurements for the ImageNet networks from Section 5 to accompany Figure 7.

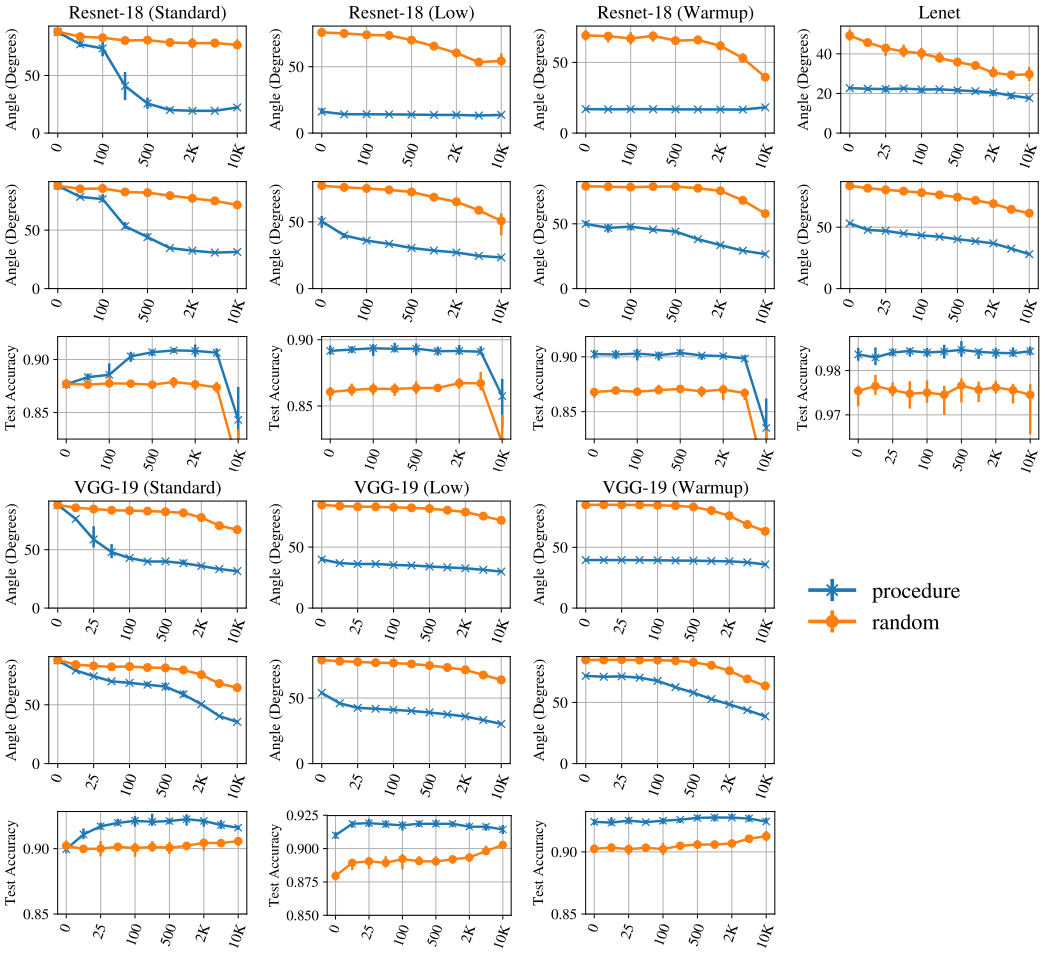

Figure 11: The effect of the rewinding iteration (x-axis) on angle data order stability (top), angle pruning stability (middle), and accuracy (bottom) for each network in Figure 2. This figure accompanies Figure 4.

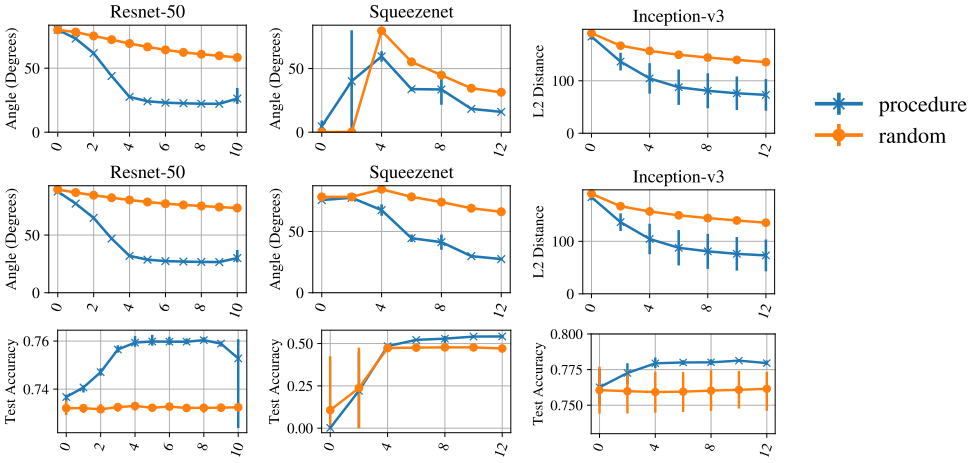

Figure 12: The effect of the rewinding epoch (x-axis) on data order stability (top), pruning stability (middle), and accuracy (bottom) for each network in Figure 6.