Lottery Tickets on a Data Diet: Finding Initializations with Sparse Trainable Networks

Mansheej Paul[1*] **Brett W. Larsen**[1*]
Surya Ganguli[1,2] **Jonathan Frankle**[3,4,5] **Gintare Karolina Dziugaite**[6,7]
[1]Stanford [2]Meta AI [3]MIT [4]MosaicML [5]Harvard [6]Google Brain [7]Mila; McGill

Abstract

A striking observation about iterative magnitude pruning (IMP; Frankle et al. 10) is that—after just a few hundred steps of dense training—the method can find a sparse sub-network that can be trained to the same accuracy as the dense network. However, the same does not hold at step 0, i.e., random initialization. In this work, we seek to understand how this early phase of pre-training leads to a good initialization for IMP both through the lens of the data distribution and the loss landscape geometry. Empirically we observe that, holding the number of pre-training iterations constant, training on a small fraction of (randomly chosen) data suffices to obtain an equally good initialization for IMP. We additionally observe that by pre-training only on "easy" training data we can decrease the number of steps necessary to find a good initialization for IMP compared to training on the full dataset or a randomly chosen subset. Finally, we identify novel properties of the loss landscape of dense networks that are predictive of IMP performance, showing in particular that more examples being linearly mode connected in the dense network correlates well with good initializations for IMP. Combined, these results provide new insight into the role played by the early phase training in IMP.

1 Introduction

Modern deep neural networks are often trained in the massively over-parameterized regime. Though these networks can eventually be pruned, quantized, or distilled into smaller networks, the resources required for the initial over-parameterized training poses a challenge to the democratization and sustainability of AI. This raises a fundamental question: under what circumstances can we efficiently train sparse networks? Recent work on the lottery ticket hypothesis [9, 10] has shown that, after just a few hundred steps of pre-training, a dense network contains a sparse sub-network that can be trained without any loss in performance. Finding this sparse sub-network currently requires multiple rounds of training to convergence, pruning, and rewinding to the pre-train point, a procedure termed iterative magnitude pruning (IMP, Figure 1; [9, 10]), Remarkably, even after all these rounds of training, we do not find trainable sparse sub-networks if we rewind to the random initialization; the first few hundred steps of dense network training is essential for finding sparse networks through IMP. In this work, we seek to understand this *very short but critical phase* of pre-training. In particular, we investigate the effect of training data and number of steps used during pre-training on the accuracy achieved by IMP. We also explain how certain properties of the loss landscape allow us to predict whether we will find a *matching initialization* (i.e., an initialization from which IMP "succeeds" in finding a subnetwork that can match the accuracy of the unpruned network, to be formalized later).

IMP proceeds in *three* phases: an initial *pre-training phase* where (1) the dense weights are trained for a few hundred steps, followed by (2) the resource-intensive *mask search phase*, during which we

*Equal contribution. Correspondence to: `{mansheej,bwlarsen}@stanford.edu; gkdz@google.com`

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

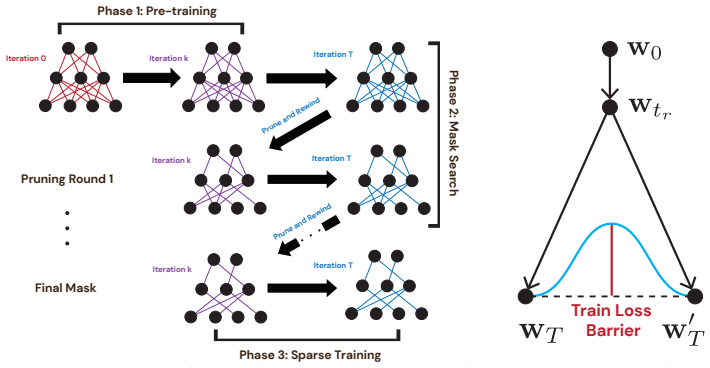


Figure 1: **Left:** Three phases of iterative magnitude pruning (IMP) with weight rewinding [10]. A dense network is trained in the pre-training phase for t_r iterations, where t_r is referred to as the rewinding iteration, and $\mathbf{w}_{t_r}$ is the parameters of the network at the rewinding point. The mask search phase produces a sparse subnetwork at a desired sparsity level by iteratively training, pruning the smallest magnitude weights, and rewinding to $\mathbf{w}_{t_r}$. The sparse training phase trains the final sparse subnetwork to convergence, starting with weights $\mathbf{w}_{t_r}$. **Right:** Illustration of computing the train loss barrier for initialization $\mathbf{w}_{t_1}$. $\mathbf{w}_T$ and $\mathbf{w}'_T$ are trained with different data order.

iteratively find a sparse mask by training the network and rewinding the weights repeatedly, finally, ending with (3) the *sparse training phase*, when the sparse masked network is trained. The weights learned after pre-training phase—which serve as the initialization for mask search phase—are referred to as the *pre-trained initialization* (see Figure 1).

In this work, we focus on studying the pre-training phase. To probe sufficient information needed to arrive at a matching initialization for IMP, we modify the pre-training phase by training with different pruned datasets and varying the number of pre-training steps; we then compare the performance of the resulting sub-networks across a range of sparsities. Following Paul et al. [23], we prune the datasets both randomly and according to EL2N scores, which estimate example difficulty by measuring early training performance in an ensemble of networks. We next turn to characterizing the loss landscape properties of matching initialization for IMP. Frankle et al. [10] investigate the relationship between successful initializations for IMP and linear mode connectivity on the *sparse* initialization for the entire dataset; here, we investigate this relationship on the *dense* initialization on a per-example basis. This enables us to identify signatures of the *dense* loss landscape correlated with a matching initialization. Thinking more broadly, we explore the relationship between sparse trainability and training stability. Gilmer et al. [13] demonstrate that, in hyperparameter regimes in which early training is unstable, learning rate warmup helps stabilize training. We build on this by investigating if the same pruned datasets which decreased the amount of pre-training required to find a matching initialization for IMP also decrease the required duration of learning rate warmup.

Overall, this empirical analysis provides new insight into the important and mysterious role that pre-training the dense network plays in IMP. Intriguingly, finding a matching initialization for IMP is an example of a problem in which the dominant paradigm of "more data, more training" is not optimal. Indeed, by carefully choosing the training examples, we can not only use a very small subset of the data but also reduce the training time required to solve this problem. From a scientific perspective, this suggests that different phases of training play different roles in the optimization process. Identifying and characterizing these phases will not only lead to a deeper understanding of the optimization of deep neural networks but also allow us to design training strategies that are optimal for each phase. From the practical standpoint, data loading is often an expensive part of the training process and can be a bottleneck especially early in training. By identifying the essential subset for this phase of training, our work may enable strategies to eliminate this bottleneck.

Contributions. We find empirical evidence for the following statements:

- **In the pre-training phase, only a small fraction of the data is required to find a matching initialization for IMP:** On standard benchmarks, across all sparsity levels we evaluated, we find that we can match accuracy by training on a small fraction of all of the available training data, selected randomly. (As we vary the amount of training data in pre-training phase, the number

of training iterations is held fixed.) Note that this observation changes if random label noise is introduced, in which case it becomes important to select easy (small EL2N score) examples.

- **The length of the pre-training phase can be reduced if we train only on the easiest examples:** Informally, training on a small subset of "easy-to-learn" training examples produces a better rewinding point than training on all data for the same number of iterations;
- **The quality of a pre-trained initialization for IMP correlates with more examples being linearly mode connected in the dense network:** This result complements the empirical evidence produced by Frankle et al. [10] connecting linear mode connectivity and the performance of IMP.
- **Training on easy data, which reduces the amount of pre-training required to find a matching initialization for IMP, does not reduce the amount of training required to stablize the network via learning rate warmup ([13]):** The role played by data during IMP pre-training is thus different than that during learning rate warmup.

2 Background, Methods, and Related Work

We consider standard neural network training on image classification. Let $S = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ denote training data, let $\mathbf{w}_t \in \mathbb{R}^D$ denote model parameters (weights) of the neural network, and let $\mathbf{w}_1, \mathbf{w}_2, \ldots$ be the iterates of (some variant) of SGD, minimizing the training *loss*, i.e., average cross-entropy loss over the training data. For a given training example $\mathbf{x}_n$, let $f(\mathbf{w}, \mathbf{x}) \in \mathbb{R}^K$ denote the logit outputs of the network for weights $\mathbf{w}$ and $p(\mathbf{w}, \mathbf{x}) = \sigma(f(\mathbf{w}, \mathbf{x}))$ be the probability vector returned by passing the logits through the softmax operation σ. By the *loss (error) landscape*, we mean the training loss (error), viewed as a function of the parameters. By training and test *error*, we mean the average 0-1 classification loss.

Lottery ticket subnetworks. The *lottery ticket hypothesis* [9] states that any standard neural network "contains [at initialization] a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations." Although such *matching subnetworks* (those that can train to completion on their own and reach full accuracy by following the same procedure as the unpruned network) are not known to exist in general at random initialization, they have been shown to exist after *pre-training* the dense network for a short amount of time (Phase 1 in Figure 1) before pruning [2, 10, 17, 25, 27].

Empirical evidence for this phenomenon comes via a procedure that finds such subnetworks retroactively after training the entire network. This procedure, called *Iterative Magnitude Pruning* [IMP; 10] is based on standard iterative pruning procedures [14], and can be decomposed into three phases (Figure 1), outlined in Algorithm 1.

Algorithm 1 IMP rewinding to step t_r and N iterations.

1: Create a network with randomly initialization $\mathbf{w}_0 \in \mathbb{R}^d$.
2: Initialize pruning mask to $\mathbf{m} = 1^d$.
3: Train $\mathbf{w}_0$ for t_r steps to $\mathbf{w}_{t_r}$ ▷ Phase 1: Pre-Training
4: **for** $n \in \{1, \ldots, N\}$ **do** ▷ Phase 2: Mask Search
5: Train the pruned network $\mathbf{m} \odot \mathbf{w}_{t_r}$ to completion. ($\odot$ is the element-wise product)
6: Prune the lowest magnitude 20% of weights after training.
Let $\mathbf{m}[i] = 0$ if the corresponding weight is pruned.
7: Train the final network $\mathbf{m} \odot \mathbf{w}_{t_r}$. Measure its accuracy. ▷ Phase 3: Sparse Training

This procedure reveals the accuracy of pre-training the dense network for t_r iterations, pruning, and training the pruned network thereafter. Phase 2 can be understood as an (expensive) oracle for choosing weights to prune at t_r. Although IMP is too expensive to use as a practical way to speed up training, it provides a window into a possible minimal number of parameters and operations necessary to successfully train a network to completion in practice. In our work, we extend this line of thinking, pursuing the minimal amount of data necessary to find and train these subnetworks. This is especially tantalizing due to the potential positive interactions between sparsity and minimizing the data necessary for training. *The result is a deeper inquiry into the minimal recipe for successful training and, thereby, into the fundamental nature of neural network learning in practice.*

In this respect, the closest work to ours is an experiment in a larger compendium by Frankle et al. [11] showing that the standard pre-training phase could be replaced by a much longer self-supervised phase.

Like our experiments in Section 3, that work aims to study what makes a matching initialization. Our approach and findings are substantially different, however: we reduce the number of examples rather than changing the labels, and we show that, not only are a small set of examples (starting at $\approx 2\%$) sufficient for pre-training, but also that they make it possible to pre-train in *fewer* steps.

There are many other ways to obtain pruned neural networks [e.g., 7, 14, 16, 20, 29]. The distinctive aspect of work on the lottery ticket hypothesis (and the one that makes it the right starting point for our inquiry) is that its goal is to uncover a minimal path from initialization to a trained network, regardless of the cost of doing so. The aforementioned procedures target real-world efficiency for training and/or inference.

Linear Mode Connectivity and the Loss Barrier. We investigate the error landscape using the parent–child methodology and instability analysis of Frankle et al. [10] and Fort et al. [8]. Writing $\mathrm{err}(\mathbf{w})$ for the test error at the weights $\mathbf{w}$, the (test) loss barrier between two networks $\mathbf{w}$ and $\mathbf{w}'$ is $\sup_{\alpha\in[0,1]}[\mathrm{err}(\alpha\,\mathbf{w}+(1-\alpha)\mathbf{w}')-(\alpha\,\mathrm{err}(\mathbf{w})+(1-\alpha)\mathrm{err}(\mathbf{w}'))]$. This quantity measures the maximum increase in loss above the average loss along the linear path connecting the two networks on the loss landscape.

The loss barrier at iteration t (with parent weights $\mathbf{w}_t$) is the loss barrier between the (children) weights $\mathbf{w}_T$ and $\mathbf{w}'_T$, where $\mathbf{w}_T$ and $\mathbf{w}'_T$ are copies of $\mathbf{w}_t$ trained to completion with the same procedure but different random seeds (minibatch order, GPU noise, data augmentation, etc.). See Figure 1 right for a visualization. Empirically, the loss barrier is achieved near $\alpha=\frac{1}{2}$, and so we compute it this way. The *onset of linear mode connectivity (LMC)* is defined to be the iteration $\bar{t}$ such that, for all $t>\bar{t}$, the error barrier at t is zero. We follow Draxler et al. [4], Garipov et al. [12], and Frankle et al. [10] in considering the 0-1 loss barrier to be zero if it is less than 2%. Similar to Paul et al. [23], we also measure the error (0-1 loss) or cross-entropy loss barrier on individual training examples to explore the loss landscape corresponding to different subpopulations.

Ranking training examples. We define *"easy/hard data"* as the data that is ranked low/high, respectively, by the EL2N score introduced by Paul et al. [23]. EL2N scores depend on the margin early in training, and, loosely speaking, higher average margin early in training means lower importance for generalization of the final trained model. This connection to margin suggests that easy data is learned first (has higher margin early in training, maintained throughout the rest of training). EL2N scores were derived from the size of the loss gradient, and are thus are highly correlated with the magnitude of the gradient.

Definition 2.1 (EL2N Score) *The EL2N score of a training sample* $(\mathbf{x},\mathbf{y})$ *at iteration* t *is defined as* $\mathbb{E}\|p(\mathbf{w}_t,\mathbf{x})-\mathbf{y}\|_2$*, where the expectation is taken over* w_t *conditioned on the training data.*

To calculate EL2N scores for a dataset, we follow the process outlined in [23]. In particular, we do the following:

1. Independently train $K=10$ networks from different random initializations for t iterations.

2. For each example and each network, we calculate the L2 norm of the error vector defined as $\|p(\mathbf{x})-\mathbf{y}\|_2$ where $\mathbf{y}$ is the one-hot encoding of the label, and $p(\mathbf{x})$ are the softmax outputs of the network evaluated on example $\mathbf{x}$.

3. For each example, the EL2N score is the average of the error vector L2 norm across the K networks.

In our experiments (Section 3), we vary the data that is accessible in the pre-training phase of IMP defined above. We either choose the data that we feed to the algorithm at random while preserving class balance, or based on the EL2N scores.

Stabilizing early training via learning rate warmup. Learning rate warmup period can be seen as a form of pre-training, allowing one to eventually train at higher learning rates and larger batch sizes. Training with large batches and high learning rates is desirable in practice, as it allows for more efficient GPU utilization and may reduce the total number of updates needed to achieve the desired accuracy. Gilmer et al. [13] empirically observe that learning rate warmup essentially improves the optimization by allowing the initial optimization trajectory to navigate to "flatter" optimization landscape, i.e., one with smaller highest loss Hessian eigenvalue.

3 The Role of Training Data Selection in Pre-Training

As can be seen in Figure 2, when training sparse networks using IMP with rewind step $t_r = 0$, the final test accuracy of the sparse networks falls off rapidly with increasing sparsity. However, as we increase the rewind step t_r, the network performance improves across all sparsity levels and at a rewind step t^*, the network performs as well as or better than the dense network at high sparsities. Informally, training the dense network for t^* steps creates a matching initialization for IMP. But what does the network learn in these first t^* steps? In this section, we take the first step towards answering this question by investigating which subsets of the training data are sufficient for finding a matching initialization. In order to compare networks trained on different subsets of data for different numbers of iterations, we introduce the notion of a *matching initialization* with the following definitions.

Definition 3.1 *Let $\mathbf{w}_t^S$ be the dense network weights after training on a subset of the training data, S until rewind step t. Then for two data subsets $\{S, S'\}$, rewind times $\{t, t'\}$ and a given range of sparsities, $\mathbf{w}_{t'}^{S'}$ is said to* dominate *(weakly dominate) $\mathbf{w}_t^S$ if sparse networks obtained from IMP with $\mathbf{w}_{t'}^{S'}$ as the initialization achieve better (no-worse) accuracy than those obtained from IMP with $\mathbf{w}_t^S$ as the initialization.*

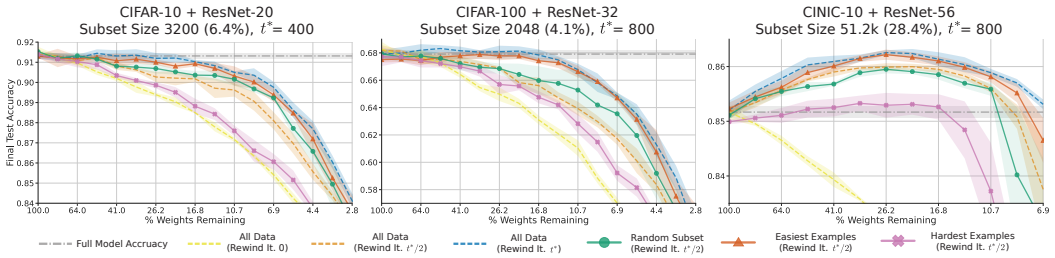


Figure 2: For a given rewind step $t_r = t^*/2$, training on a small fraction of random data during the pre-training phase of IMP leads to matching initializations (compare the solid green with circles and dashed orange curves) across dataset, network, and hyperparameter configurations. Using just the easiest training examples during this phase produces a matching initialization for rewind point t^* in just $t^*/2$ steps (compare the solid red with triangles and dashed blue curves). Pre-training on the hardest examples is detrimental to the performance of the initialialization (solid pink curve with crosses). IMP with rewinding to initialization (dashed yellow curve) and the dense model (dashed grey curve) are used as baselines. For each dataset + network configuration, we present the best performing easy data subset size. For a sweep across subset sizes, see Figure 3 and the Appendix E.

For a network trained on the full dataset for t steps, we write $\mathbf{w}_t$. In Figure 2, we see that $\mathbf{w}_{t^*}$ dominates $\mathbf{w}_{t^*/2}$ which in turn dominates w_0. We investigate which data subsets S and rewind steps t lead to networks $\mathbf{w}_t^S$ that dominate $\mathbf{w}_{t^*}$ and $\mathbf{w}_{t^*/2}$—such networks are called matching initializations.

Definition 3.2 *A dense network $\mathbf{w}_t^S$ is a matching initialization for rewind time t^* if $\mathbf{w}_t^S$ weakly dominates $\mathbf{w}_{t^*}$.* [2]

We empirically find that certain surprisingly small subsets S and rewind step $t_r < t^*$ lead to matching initializations for rewind time t^*.

Experimental design. To evaluate the effect of the training subset size and composition on the quality of the pre-trained initialization, we train ResNet-20/ResNet-32/ResNet-56 on subsets of CIFAR-10/CIFAR-100/CINIC-10, respectively. The subset size M is varied and subsets are chosen as follows: (i) M randomly selected examples, distributed equally among all classes; (ii) the easiest M examples; (iii) the hardest M examples. The easiest examples are those with the smallest EL2N scores and the hardest are the examples with the largest EL2N scores [23].

[2] Note that due to the stochasticity in training and risk measurements, we ignore small deviations in the final test accuracy. Thus one rewinding point could weakly dominate another one even if at some sparsity levels their mean performance "crosses" over while approximately remaining within the standard error of one another.

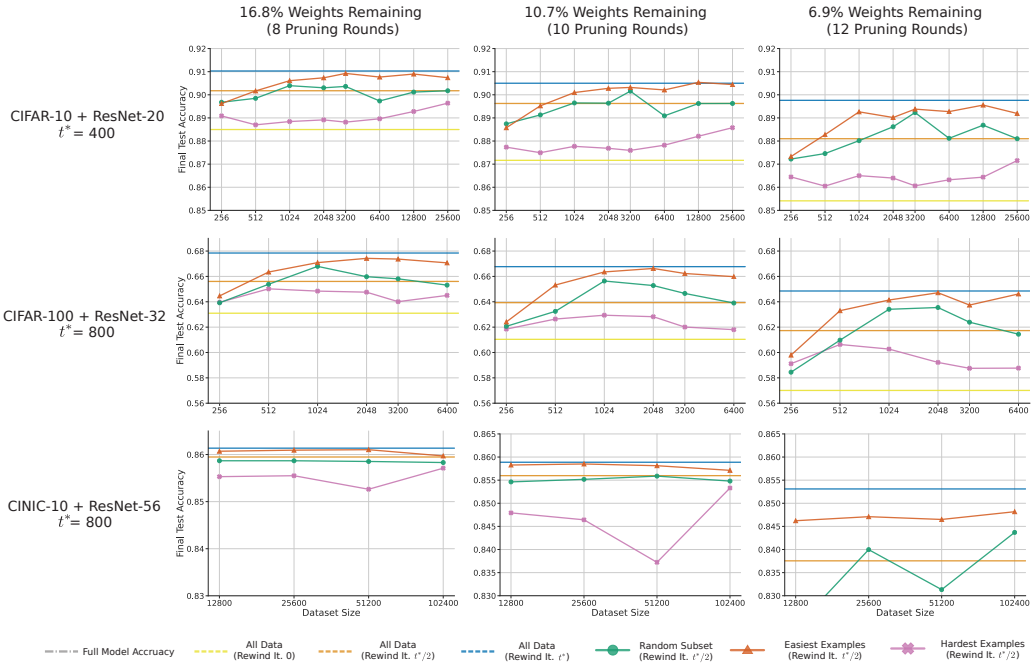


Figure 3: A summary of the the dependence on subset size for the style of experiments described in Figure 2. The first column represents the performance across subset size for the fixed sparsity 16.8% weights remaining or 8 rounds of pruning. The subsequent columns show the same for 10.7% (10 pruning rounds) and 6.9% (12 pruning rounds) respectively. The horizontal lines correspond to baseline runs at rewind steps 0, $t^*/2$, and t^* using all the data. For CINIC-10 (bottom row), rewind step 0 and the hardest data subsets are not visible in some cases because their accuracies fall below the range displayed.

Due to the significant computational demands of performing IMP with multiple pre-training schemes and replicates, we focus on a targeted set of pre-training iterations t_r. In particular, we study the pre-training iteration $t_r = t^*$ where training on all examples leads IMP to find spare sub-networks that perform as well as the dense network for a large range of sparsities (t^* = 400 for CIFAR-10 and 800 for CIFAR-100 and CINIC-10). We also study the more challenging pre-training iteration of $t_r = \frac{t^*}{2}$, where pre-training on all data does not yield a matching initialization. When M examples are not enough to train for t_r iterations without replacement, we make multiple passes over the M examples as necessary. Figure 2 shows the best performing easy data subset for each dataset across the full range of sparsities; Figure 3 shows the performance across subset size at three fixed sparsities.

Randomly chosen examples. Pre-training the dense networks on small, randomly chosen subsets S can lead to initializations for IMP, $\mathbf{w}_{t_r}^S$, that dominate initializations $\mathbf{w}_{t_r}$, trained on the entire training set for the same number of steps. In Figure 2 we see that for all dataset + network combinations, pre-training the dense network on a small random subset (solid green curve with circles; sizes ranging from 4.1% for CIFAR-100 to 28.4% for CINIC-10) for $t_r = t^*/2$ leads to initializations that (weakly) dominate those that were obtained from training the network for the same number of steps on all the data. This observation leads to a surprising suggestion: in these experiments, the subset size is smaller than the total number of images seen during the pre-training phase; for the particular goal of finding a matching initialization of IMP, multiple passes through the same small dataset can be more beneficial than seeing more random data.

Easiest examples (lowest EL2N scores). By pre-training on just the easiest examples (identified by lowest EL2N scores, solid red curve with triangles in Figure 2), we can obtain matching initializations in fewer steps compared to training on the full dataset. In Figure 2, we see that for all three dataset and network combinations and for the subset sizes shown, the initialization obtained from training on the easiest examples for $t_r = t^*/2$ steps leads to matching initializations for t^*.

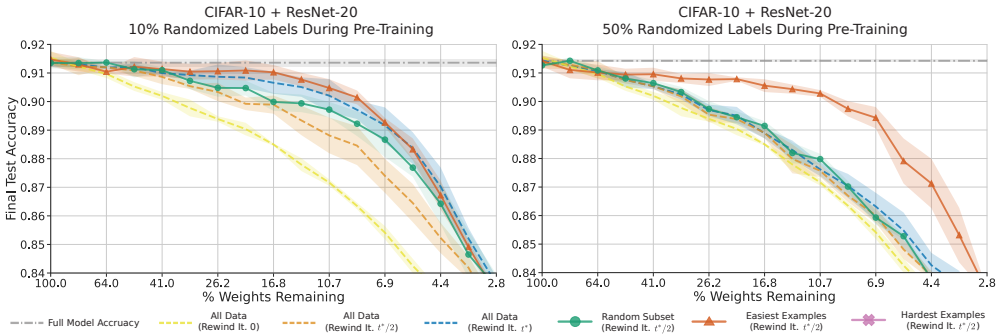


Figure 4: Pre-training on a random subset or all data is not robust to label noise during this initial phase of IMP. However, pre-training with the easiest data as scored by EL2N scores computed from the corrupted dataset is robust. In both the left (10% randomized labels) and right (50% randomized labels), pre-training on the easiest data for $t^*/2$ iterations dominates all other pre-training schemes, including training on all data for t^* iterations. Results for additional subset sizes are included in Appendix E.

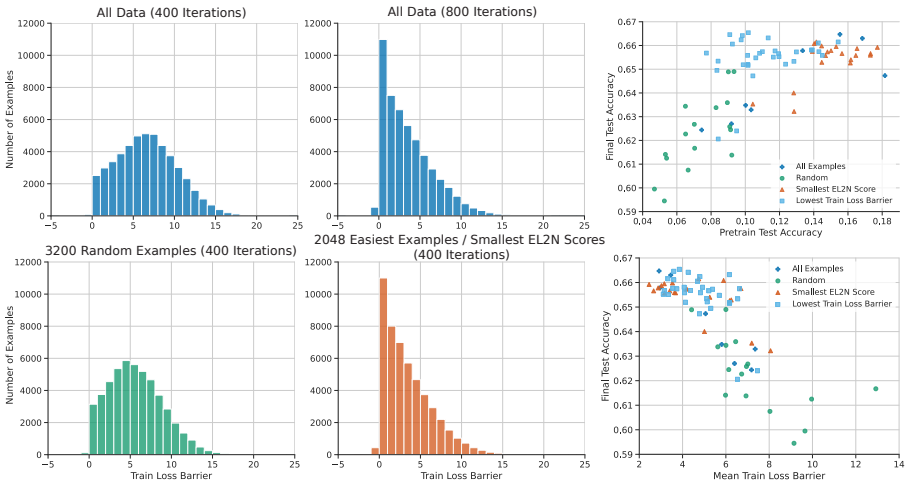
Hardest examples (highest EL2N scores). Conversely, pre-training on the hardest examples (solid pink curve with crosses in Figure 2) yields worse accuracies than pre-training on all examples or a random subset. In fact, on CIFAR-10 the hardest examples perform little better than using no pre-training at all. Interestingly, when training a dense network, these hardest examples are crucial for obtaining a network with good generalization properties [23]. This suggests that while the hard example may be key later in training, repeated passes through easier examples should be the focus during the very early stages of training to quickly find a good initialization for IMP.

Randomized labels during pre-training. Pre-training on a all data or a random subset is not robust to corruption with random label noise [28] during the pre-training phase. As seen in Figure 4, the higher the percentage of randomized labels, the lower the performance of these data subsets, and in particular, the pre-trained rewinding point becomes no better than a random initialization when 50% of the labels are randomized. On the other hand, training on easiest data with EL2N scores computed on the corrupted dataset is robust to this noise (solid red curve with triangles in Figure 4). This is because examples with randomized labels are hard ([23]) according to this metric, and thus the easiest examples will select a subset of largely uncorrupted data.

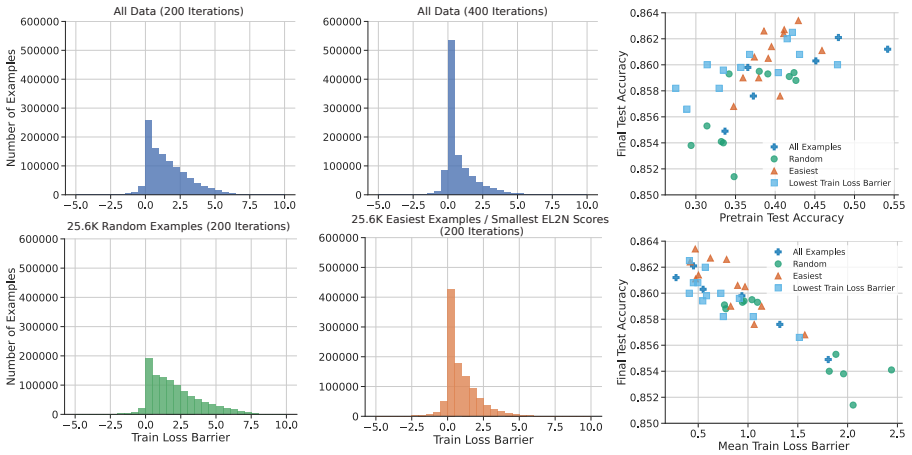
Summary. Taken together, our results suggest that, finding a matching initialization for IMP at rewinding step t^* is an interesting problem in which "more data, more training" is *not* optimal; it is neither necessary to train on all the data nor to train for the full t^* steps. In fact, we can get away with training on a surprisingly small dataset for as little as half the number of steps if we make multiple passes through the *right* examples, here the easiest examples as defined by lowest EL2N scores.

4 Pre-training through the lens of Linear Mode Connectivity

Frankle et al. [10] observed a strong indicator of whether a *sparse* sub-network generated by IMP would be able to match the accuracy of the dense network is whether or not that sparse sub-network train to the same linearly connected mode (i.e. the train loss barrier is close to 0, see Figure 1). Here we ask what properties of the *dense* network at the rewind point might also be predictive of this property. In the architectures + datasets we consider, the onset of linear mode connectivity (LMC) occurs in the dense network later than the first rewind time which produces a good initialization for IMP (i.e. $\bar{t} > t_r$), and thus, we consider two generalizations to the notion of LMC. First, we look at the train loss barrier on a per-example basis meaning that instead of considering the full loss landscape we separately look at the performance of the linear interpolation on each example (the original notion of train loss barrier is obtained by averaging these values). Second, we then look at the distribution of these per-example train loss barriers as a continuous measure of the state of the network rather than simply considering whether their average is 0. We demonstrate in general that these measures of the loss landscape of the dense network correlate well with the IMP performance of the pre-trained initialization. We additionally show that pre-training on easy data results in a



(a) CIFAR-100, ResNet-32, $t^* = 800$ and $t_r = t^*/2 = 400$.



(b) CINIC-10, ResNet-56, $t^* = 400$ and $t_r = t^*/2 = 200$.

Figure 5: **Left and center columns:** Pre-training on all the data for t^* (top right) vs. $t^*/2$ (top left) steps leads to a distribution shift to smaller per-examples train loss barriers between children runs of the dense network. When pre-trained for just $t^*/2$ iterations but with a random subset of data, the distribution is similar to pre-training on all data for $t^*/2$ steps. However, when pre-trained on a subset of easy examples for $t^*/2$ iterations, the distribution of train loss barriers displays a shift comparable to pre-training on all the data for t^* iterations. As seen in Figure 2 and the Appendix, pre-training with this random subset matches pre-training with all data for $t^*/2$ steps while pre-training with the easiest data for $t^*/2$ matches pre-training on all data for t^* steps. This observation suggests that the per-example distribution of train loss barriers may be a useful signature for predicting the IMP performance of dense initializations. **Right column:** Scatter plots of final test accuracy vs. pre-train accuracy of the dense initialization (top) and the mean train loss barrier (bottom) across a variety of dataset pruning strategies for pre-training. Final test accuracy is at 8.6% weights remaining (11 pruning rounds) for CIFAR-100 and 21% weights remaining (7 pruning rounds) for CINIC-10. We observe that the mean train loss barrier is better correlated with the final test accuracy than the pre-train accuracy of the dense initialization.

distribution with a large fraction of examples linearly mode connected faster than pre-training on all data. See Appendices B and C for additional experiments and an exploration of why this occurs.

The distribution of per-example loss barriers. In the left and center columns of Figure 5 we plot histograms of per-example loss barriers for various rewinding points and data subsets for CIFAR-100 (panel a) and CINIC-10 (panel b). There is a clear shift in the empirical distribution of the per-example loss barriers — not only the mean is shifting as previously observed, but also the mode. Pre-training

on all data for t^* iterations has the same effect on the distribution of the loss barriers as pre-training on the easy examples for $t^*/2$ iterations, two procedures which produce matching IMP initializations.

Given the connection between the error barrier and IMP sub-network performance, it is natural to ask whether examples which achieve low loss barrier early in training with the full dataset are more important for producing a matching initialization than high loss barrier examples. We call this ranking of the data by the per-example loss barrier at a given iteration the LMC score, and the blue squares in the scatter plot are the result of training on a subset of the data determined by this score. Our empirical findings suggest that using this score does not match the performance of using easy data (see Appendix D).

The correlation between final test accuracy and mean train loss barrier. The right column of Figure 5 shows a scatter plot of final test accuracy of the sparse trained network vs. two different properties of the dense initialization: test accuracy of this pre-train point (top) and the average train loss barrier from 3 pairs of children runs (bottom). Here we observe the same result across the different training conditions: a correlation between the train loss barrier when spawning from w_{t_r}, and the final test accuracy of the sparse sub-networks. The correlation is weaker for the pre-train test accuracy, suggesting that the loss landscape properties hold more import for determining the success of IMP. Furthermore, we see that training on the easiest examples produces rewinding points with smaller train loss barriers than training on random subsets.

5 The Role of Training Data in Learning Rate Warmup

To this point, we have studied the early phase of training through the lens of the lottery ticket hypothesis. However, the dynamics of the early phase of training have important implications beyond the sparse regime considered by lottery ticket research. For example, learning rate warmup is a nearly ubiquitous part of training state-of-the-art models. Warmup can be viewed as an accommodation for the idiosyncrasies of the earliest part of training and a pre-training strategy that prepares the network to train at the full learning rate. In this section, we extend our previous experiments to dense training with learning rate warmup by ablating the training data during this phase to assess whether easy data alone suffices, similar to our observations in the pre-training phase for IMP (Section 3).

Easy data does not reduce the amount of training required to stabilize the network via learning rate warmup. The effects of performing learning rate warmup with easy, random, and hardest subsets of the data appear in Figure 6. Intriguingly, the most striking performance drop is observed when doing learning rate warmup with easy data, especially at longer pre-training times. Pre-training on the random and hardest data produces the same performance for sufficiently large subset sizes. The role played by data during IMP pre-training is thus different than that during learning rate warmup; the easy data does not add stability benefits during the later. We hypothesize that this is because learning rate warmup happens over a much larger fraction of the total training time than IMP pre-training.

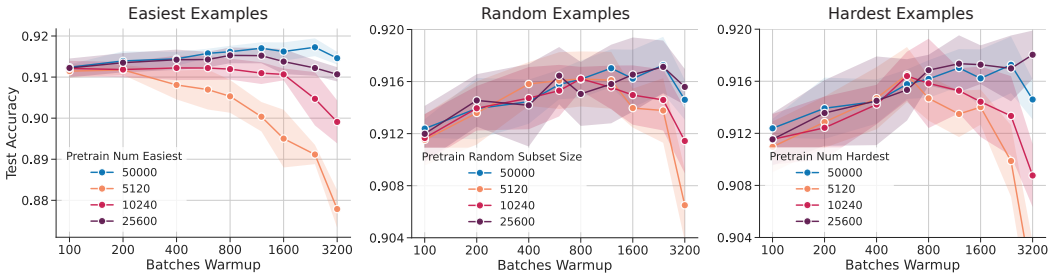


Figure 6: Training with different data subset sizes and pruning strategies for the learning rate warmup period for ResNet-20 on CIFAR-10. In the large batch size and learning rate regime, learning rate warmup is essential for stabilizing training; we sweep over different learning rate warmup periods with a batch size of 1024. In contrast to pre-training for IMP, training on easiest examples in this initial phase leads to worse performance than training with all data, random subsets, or the hardest data. We note that the learning rate warmup period is a much longer fraction of training than IMP pre-training which lasts for only a few hundred steps.

Discussion and algorithmic implications. The benefits of training on easy data—faster learning and increased stability to SGD noise—are only present in the very early phase of learning. Paul et al. [23] have shown that, if we continue to train on easy data only, the network will converge to a lower final test accuracy. So a natural question is what other scenarios do the benefits of training on easy data extend to? We began to investigate this question here by studying training at high learning rates where learning can become unstable early in training and is usually stabilized by learning rate warmup [13]. Our results show that training only on easy examples does not help and in fact hurts performance in this case. Two possible reasons for this behavior are (1) learning rate warmup is often used for the first 10% of training, which may be too long for easy examples to be helpful (the pre-training phase in iterative magnitude pruning where easy examples are clearly beneficial can be as little as 2% of training) and (2) the instability in this case—loss diverging to infinity—is not reduced by methods that improve stability to SGD noise. Despite the negative results of this experiment, we think that, when combined with our previous results, the core scientific findings in our work are interesting and broadly useful to the community. In particular, some algorithmic implications are as follows: (1) While developing algorithms for training sparse networks or finding matching networks at initialization, researchers might want to incorporate methods or search for initializations that improve stability to SGD noise. (2) Recent work in curriculum learning found that curricula provide marginal benefits unless there are additional constraints such as a training budget [26]. Our work helps narrow the time window in which training on easy examples is beneficial and provides another example in which, for the constrained scenario of finding matching sparse networks, a simple curriculum of training on easy data first is beneficial. Thus, these findings may help in the design of better training curricula. (3) In the training of large language models, a common problem is that loss spikes and training becomes unstable. A hypothesis for why this phenomenon occurs is that the network is presented with a bad batch or a set of bad batches that destabilize training. Since this is related to instability due to SGD noise (bad batches are randomly sampled) it may be possible to improve training stability by injecting easy examples into the training process.

6 Discussion

Recent empirical evidence has shown that deep neural network optimization proceeds in several distinct phases of training [8, 11]. Understanding the role that data plays in these different phases can help us characterize what is being learned during them. Since data loading is often a bottleneck, this understanding also has the potential enable more efficient training schemes. In this work, we have considered two essential phases of early training: pre-training for IMP which enables sparse optimization and learning rate warmup which stabilizes network training. Our experiments identify what data is sufficient for each of these procedures, and as they both occur early in training, these finding can be used to design more efficient data loaders for streaming datasets. Furthermore, as IMP is a computationally expensive procedure, understanding how it works is essential for designing better algorithms with the same performance. To this end, we identify loss landscape properties of the dense network initialization for IMP that are predictive of successful spare training. Though this work does not provide an improved algorithm for obtaining sparse networks, we believe our results provide essential guidance for researchers pursuing algorithms that perform pruning early in training (i.e. finding sparse masks without training to convergence).

Acknowledgements

The experiments for this paper were funded by Google Cloud research credits. S.G. thanks the James S. McDonnell and Simons Foundations, NTT Research, and an NSF CAREER Award for support while at Stanford. This work was done in part while G.K.D. was visiting the Simons Institute for the Theory of Computing. The authors would like to thank Daniel M. Roy for feedback on multiple drafts.

References

[1] R. Baldock, H. Maennel, and B. Neyshabur. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34, 2021.

[2] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, Z. Wang, and M. Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in Neural Information Processing Systems*, 2020.

[3] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[4] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, pages 1309–1318. PMLR, 2018.

[5] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/draxler18a.html`.

[6] R. Entezari, H. Sedghi, O. Saukh, and B. Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*, 2021.

[7] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[8] S. Fort, G. K. Dziugaite, M. Paul, S. Kharaghani, D. M. Roy, and S. Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *arXiv preprint arXiv:2010.15110*, 2020.

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

[10] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proc. Int. Conf. Machine Learning (ICML)*, 2020.

[11] J. Frankle, D. J. Schwab, and A. S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkl1iRNFwS`.

[12] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] J. Gilmer, B. Ghorbani, A. Garg, S. Kudugunta, B. Neyshabur, D. Cardoze, G. E. Dahl, Z. Nado, and O. Firat. A loss curvature perspective on training instabilities of deep learning models. In *International Conference on Learning Representations*, 2021.

[14] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] S. A. Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989.

[17] N. M. Kalibhat, Y. Balaji, and S. Feizi. Winning lottery tickets in deep generative models. *arXiv preprint arXiv:2010.02350*, 2020.

[18] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55:5, 2014.

[19] R. Kuditipudi, X. Wang, H. Lee, Y. Zhang, Z. Li, W. Hu, R. Ge, and S. Arora. Explaining landscape connectivity of low-cost solutions for multilayer nets. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605, 1990.

[21] K. Mangalam and V. U. Prabhu. Do deep neural networks learn shallow learnable examples first? In *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019. URL `https://openreview.net/forum?id=HkxHv4rn24`.

[22] V. Nagarajan and J. Z. Kolter. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] M. Paul, S. Ganguli, and G. K. Dziugaite. Deep learning on a data diet: Finding important examples early in training. *arXiv preprint arXiv:2107.07075*, 2021.

[24] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[25] M. A. Vischer, R. T. Lange, and H. Sprekeler. On lottery tickets and minimal task representations in deep reinforcement learning. *arXiv preprint arXiv:2105.01648*, 2021.

[26] X. Wu, E. Dyer, and B. Neyshabur. When do curricula work? In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=tW4QEInpni`.

[27] H. Yu, S. Edunov, Y. Tian, and A. S. Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1xnXRVFwH`.

[28] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.

[29] M. Zhu and S. Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 1
   (b) Did you describe the limitations of your work? [Yes] See Section 6, as well as Section 3 (Experimental design).
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We do not see any negative societal impact, as our work provides better understanding of existing algorithms.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix A for a link to the code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix A.
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] It does not.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental Details

Code. The code used to run the experiments is available at: `https://github.com/mansheej/lth_diet`

Datasets. We used CIFAR-10, CIFAR-100 [18], and CINIC-10 [3] in our experiments. For CINIC-10, we combine the training and validation sets into a single training set with 180,000 images. The standard test set of 90,000 images is used for testing. Each dataset is normalized by its per channel mean and standard deviation over the training set. All datasets get the same data augmentation: pad by 4 pixels on all sides, random crop to 32×32 pixels, and left-right flip image with probability half.

Models. In these experiments we use ResNet-20, ResNet-32, ResNet-56 [15]. These are the low-resolution CIFAR variants of ResNets from the original paper. The variants of the network used are specified in the figures.

Randomized Labels. The labels were randomized during the pre-training phase *only* by first selecting 10%/50% of the training uniformly at random and then drawing a new label uniformly from the 10 classes of CIFAR-10. Note that because this procedure can result in an example being reassinged the correct label, on average only 9%/45% of the labels are corrupted by this procedure. The dataset is corrupted once and then resused across all subset sizes and replicates.

The EL2N scores used to determine the subset of easiest data are computed by training on the corrupted dataset. As seen in Figure 5 of [1], the network typically learns the correct label early in training for corrupted data points. As a result, the corrupted examples will be ranked as difficult by the EL2N scores as verified in [23]. Thus, the noisy labels are filtered out by pre-training on the examples with the lowest EL2N scores.

Linear mode connectivity procedure. To determine if two children of a given network are linearly mode connected, we perform the following procedure:

1. Spawn two children initialized with the current weights of the given network.
2. Independently train the two networks with different SGD noise realized through different data orders.
3. Calculate the mid-point in weight space between the two networks.
4. Calculate the training loss for the resulting network at the mid-point.
5. Calculate the training loss barrier—the loss calculated at the mid-point minus the mean of the losses calculated at the endpoints.

Train loss barriers are calculated both on a per-example basis and as averages over the whole dataset. The per-example training loss barrier is just the training loss barrier evaluated on that example. In practice, we train three children and calculate the train loss barrier between each pair and average. Following [10], we say that two networks are linearly mode connected if the train loss barrier between them is less than 2%.

EL2N score computation To calculate EL2N scores for a dataset, we follow the process outlined in [23]. In particular, we do the following:

1. Independently train $K = 10$ networks from different random initializations for t iterations.
2. For each example and each network, we calculate the L2 norm of the error vector defined as $\|p(\mathbf{x}) - \mathbf{y}\|_2$ where $\mathbf{y}$ is the one-hot encoding of the label, and $p(\mathbf{x})$ are the softmax outputs of the network evaluated on example $\mathbf{x}$.
3. For each example, the EL2N score is the average of the error vector L2 norm across the K networks.

To calculate these scores, we use ResNet-20 and $t = 7800$ iterations for CIFAR-10, ResNet-32 and $t = 7800$ iterations for CIFAR-100, and ResNet-56 and $t = 8000$ iterations for CINIC-10.

Hyperparameters. Networks were trained with stochastic gradient descent (SGD). t^* was chosen for each dataset such that $t_r = t^*$ produces sparse networks that can be trained to the same accuracy as the dense network to at least 16.8% sparsity (8 pruning levels). The pre-training learning rate was chosen based on which of the set $\{0.1, 0.2, 0.4\}$ produced the best performance at t^*. The full hyperparameters are provided in Table 1.

Table 1: Hyperparameters Used for Experiments.

	CIFAR-10	**CIFAR-100**	**CINIC-10**
ResNet Variant	ResNet-20	ResNet-32	ResNet-56
Batch Size	128	128	256
Pre-training Learning Rate	0.4	0.4	0.1
Learning Rate	0.1	0.1	0.1
Momentum	0.9	0.9	0.9
Weight Decay	0.0001	0.0001	0.0001
Learning Rate Decay Factor	0.1	0.1	0.1
Learning Rate Decay Milestones	31200, 46800	31200, 46800	15625, 23440
Total Training Iterations	62400	62400	31250
IMP Weight Pruning Fraction	20%	20%	20%

All results are reported as the mean and standard deviation of 4 replicates. The only exception is Figure 5 scatter plots where each point corresponds to 1 replicate.

Learning rate warmup experimental design. For the learning rate warm-up experiments, we train ResNet-20 on CIFAR-10 with batch size = 1024. For the optimizer, we use SGD with learning rate = 3.2, momentum = 0.9 and weight decay = 0.0001. There is an initial linear learning rate warm-up for t batches. The network is then trained for a total of 160 epochs with a LR drop by a factor of 10 at 80 and 120 epochs. In Figure 6, we sweep the learning rate warmup period t. For each hyperparameter configuration, we report the mean and standard deviation of 8 replicates. In addition to all the training data, we perform experiments with different subsets of the training data with sizes 5120, 10240 and 25600. In these cases, the network was trained on the data subset during the warm-up period only and on the full dataset afterwards. For the easiest (hardest) examples, we choose examples with the lowest (highest) EL2N scores which were calculated as described above.

Compute Resources. The experiments were performed on virtual Google Cloud instances configured with 4 NVIDIA Tesla A100 GPUs. Each experiment replicate was run on a single A100 GPU. The approximate compute time for a full run of IMP was 8 hours for CIFAR-10, 12 hours for CIFAR-100, and 10 hours for CINIC-10.

Ethical and societal consequences. Being empirically-driven work, the experiments performed consumed considerable energy. However, we view our work as a step towards the long-term goal of improving the efficiency of training neural network which will ideally lead to an increase in the democratization and sustainability of AI. Towards this aim, our paper provides insight into the role of data in early training and a better understanding of which properties of pre-trained initializations enable sparse optimization. We hope this will inspire better algorithms for training sparse networks.

B Explanation of the Role of Easy Data

Our primary observation in Section 3 is that training on a small fraction of easy data can reduce the number of steps needed to find an initialization which contains a matching sparse network, i.e. one that can be trained to the same accuracy as the dense network. An explanation for this phenomenon can be divided into two questions. First, what characteristic of a pre-trained dense network can predict if it contains a matching sparse network? Second, does training only on easy examples allow us to find networks with this characteristic faster and why?

In Section 4, we try to provide a linear mode connectivity (LMC) based explanation for the first question. Previous work [10] has shown that the rewind step at which sparse networks become matching correlates with the onset of LMC for the sparse network. Intuitively, this suggests that a notion of stability is important for finding matching sparse networks—such networks exist at the

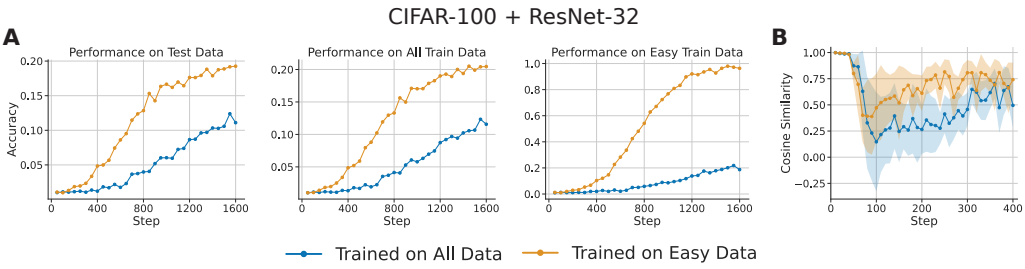


Figure 7: (a) Comparison of the performance of a network trained on all data and easy data during the first 1600 steps of training. Training on easy data results in superior test accuracy as well as train accuracy on both the full dataset and the easy subset. (b) Comparison of gradient stability for a network trained on all data and easy data. Every 10 iterations, we sample 16 batches and plot the cosine similarity between every batch's gradient and the mean gradient across batches. We see that this measure of similarity is consistently higher when trained on easy data.

rewind step at which the sparse network optimization becomes stable to SGD noise. However, this occurs before the onset of LMC in the dense network. Thus, we generalized the notion of stability to SGD noise from a binary metric—is the train loss barrier below some threshold—to a continuous one—the average train loss barrier—and found that this characteristic of the dense network is indeed correlated with our ability to find matching initializations.

To answer the second question, we show that training only on easy examples finds networks with lower average train loss barriers faster and the resulting distribution of per-example train loss barriers is very similar to those obtained from training on all data for twice as many steps (Section 4, Figure 5). Here we further investigate why this might be the case. First, we compare how fast networks learn when trained on easy vs. all data. For CIFAR-100 + ResNet-32, we train for 1600 steps on both a subset of 2048 easy examples and on all examples. We measure the test accuracy as well as the train accuracy on both the full dataset and the easy subset. The results in Figure 7a show that training on easy examples performs better on all 3 metrics, leading to faster learning. Next we compare the coherence of gradients along these training trajectories. Every 10 steps, we sample 16 batches from the dataset used to train the network and calculate the mini-batch gradients. For each step, we then measure the cosine similarity between each sampled gradient and the mean gradient across batches. The results in Figure 7b show that this similarity is consistently higher when training on easy data.

Previous work has shown that networks learn easy examples before learning hard examples [21]. Our results further show that, very early in training, making repeated passes through the easy data creates a stronger and more coherent training signal which leads to faster learning. An intuitive summary of these results is, given that the information necessary for finding matching subnetworks is learned in the first few hundred steps, it is more efficient to focus training resources on examples that can be learned well early in training.

C Other Hypotheses for the Role of Easy Data: Negative Results

Pre-training on easy data (lowest EL2N scores) allows us to reduce t_r without hurting test accuracy of the sparse networks found by IMP. One possible reason for this is that networks pre-trained on just the easy data achieve better test accuracies at step t_r. But in Figure 5, we find that the accuracy of the dense network at the pre-trained initialization does not completely explain the improved performance. Here we explore two additional hypotheses we considered for what role the easy data plays during the pre-training phase that we found to be not explanatory.

C.1 Changes in the gradient size.

1. Train a ResNet-20 on CIFAR-10 for 200 iterations on a random subset of 25600 examples. At every iteration record the gradient norm of the batch.
2. Repeat the above process but training on a subset of 25600 examples with the smallest EL2N scores.
3. Compare the distribution of gradient norms

Examples with lower EL2N scores have smaller gradient norms at the point at which they are computed as observed by [23]. We investigate whether the gradient norm is also smaller when training on the easiest examples during the pre-training phase of IMP, which would reproduce the effects of gradient clipping which is often beneficial early in training. We compare the distribution of batch gradient norms seen in the first 200 steps while training on easy data versus a random subset of data. Note that since we are only training for 200 steps with a batch size of 128, training on all the data is equivalent to training on random data. We run the following experiment:

Figure 8 shows the results. We find that, when training on easy examples, the batch gradient norms are typically larger. Our hypothesis is that on easy examples, the gradients are more aligned, leading to larger average gradients. We thus reject the hypothesis that pre-training on the easy subset of the data effectively performs gradient clipping.

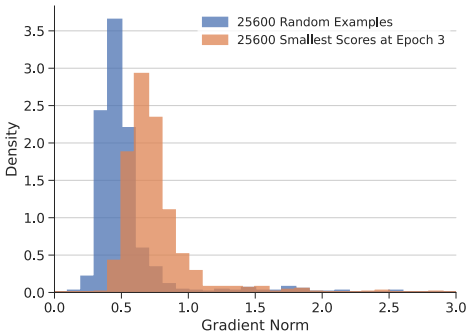


Figure 8: The histogram of minibatch gradient norms during the first $t_r = 200$ iterations of training (CIFAR-10, ResNet-20), when the minibatches are sampled from a random subset of 25600 examples compared to sampled from a subset of the 25600 easiest examples.

C.2 Change in distribution of per-example error

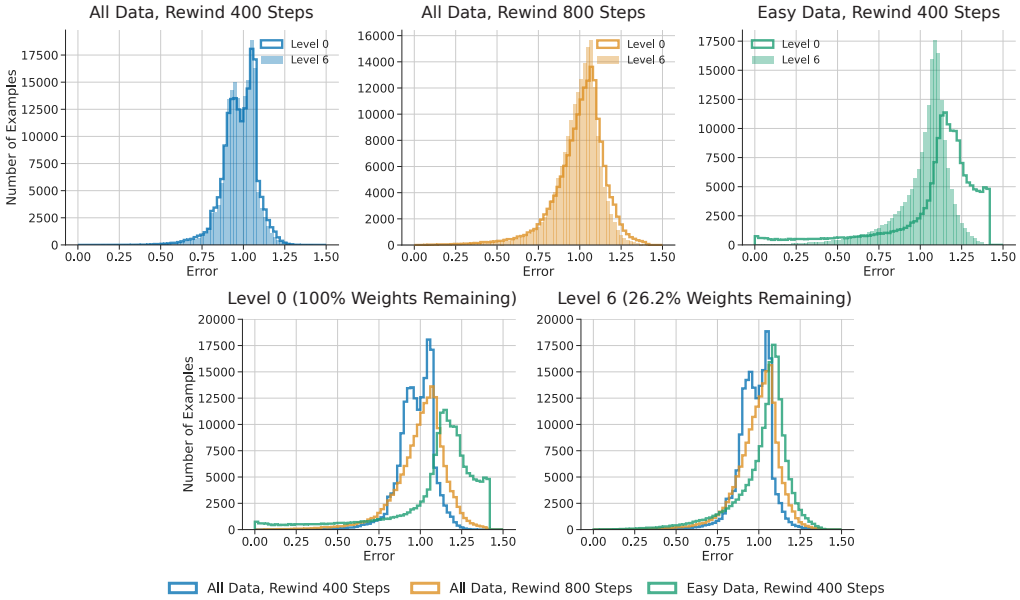


Figure 9: The distribution of per-example error at the pre-training point t_r for several different pre-training procedures before and after sparse projection. These errors are calculated for a single ResNet-56 network on CINIC-10 (i.e. not averaged across replicates). The top row shows the distributions grouped by pre-training procedure while the bottom row shows all error distributions in the dense networks (left) and sparse networks (right).

For a ResNet-56 trained on CINIC-10, we considered whether different pre-training procedures resulted in diffferent distributions of the per-example error early in training. We made the following observations in the results shown in Figure 9

1. Prior to the sparse projection, there is a distinctive distribution of error after training on easy data. A split occurs in the data where a portion of the examples shifts towards zero error while the remainder increases in error as compared to training on all data (level 0 green curve vs. level 0 blue curve in Figure 9).
2. When comparing the dense vs. sparse networks pre-trained on all data, the distribution looks the same before and after projection (level 0 vs. level 6 blue and orange curves). However, for pre-training on the easy data a significant change happens after the projection (level 0 vs. level 6 green curves).
3. After the projection, the error distributions from training on all data for 800 iterations and easy data for 400 iterations are similar (level 6 orange and green curve) while the training on all data for 400 iterations is different (level 6 blue curve).

These observations bear similarity to the per-example train loss barrier results considered in the main text (Figure 5). Pre-training on easy data for 400 iterations produces results that have the same signatures as pre-training on all data for 800 iterations (and differ from training on all data for 400 iterations). However, the per-example train loss barrier provides a clearer explanation and thus we think is the more fundamental quantity.

D Linear Mode Connetivity: Additional Experiments

Here we present additional work on understanding the connection between linear mode connectivity of a dense initialization and its performance as a pre-trained initialization for IMP. We introduce a new score for ranking examples in terms of the per-example train loss barrier early in training which we call the LMC score. We then compare this score to EL2N and test the IMP performance of pre-training on the subset of examples with low LMC scores.

D.1 Background on (linear) mode connectivity.

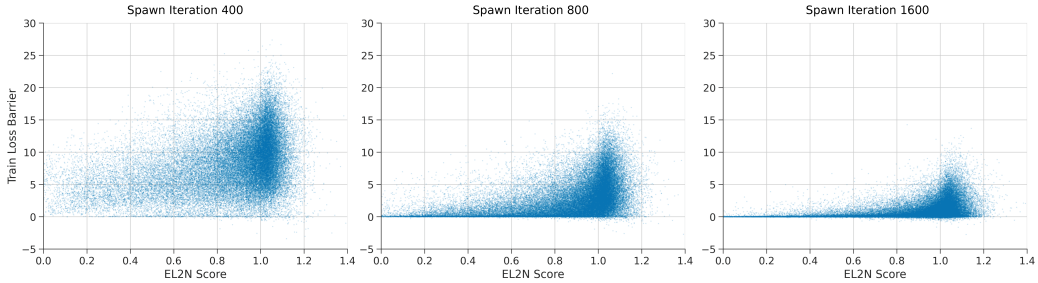
The deep learning phenomena of *mode connectivity* via simple paths has been uncovered by Draxler et al. [5], Garipov et al. [12]. In particular, it was shown that SGD-trained deep neural network modes are connected via simple nonlinear paths, along which the training and test loss is approximately constant. Since then this phenomena has also been studies theoretically [19]. A related mode property has been described in the literature on generalization in deep learning: Nagarajan and Kolter [22] show that for small networks trained on MNIST there exist *linearly* connected modes. Surprisingly, a slightly modified version of this phenomena is also true for large scale vision networks: Frankle et al. [10] show that coupling the first training epochs restricts SGD to converge to the same linearly connected mode. More recently, Entezari et al. [6] hypothesize that linearly connected modes are equivalent up to symmetries. Their empirical observations suggest that there is overlap between the functions parameterized by different modes.

D.2 LMC Scores

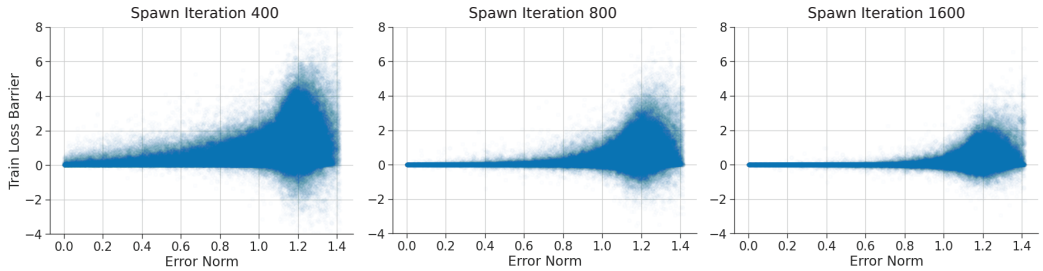
Our observation that successful pre-train initializations are correlated with a shift to smaller per-example train loss barriers (left and center columns of Figure 5) naturally raises the question: are examples that become linearly mode connected first also the ones which are important for this phase of training? To this end, we introduced the notion of a per-example LMC score which allows us to rank examples in this manner:

Definition D.1 *The* LMC score *of an example is the loss barrier computed on a dense network at rewinding iteration* t *averaged over* K *instances.*

Under this definition, the train error barrier is the LMC scores averaged over the training data. If we compute the score after the onset of linear mode connectivity $\bar{t}$ it will not provide an informative ranking of the data as the score will be close to 0 for all examples; we compute the scores before $\bar{t}$.

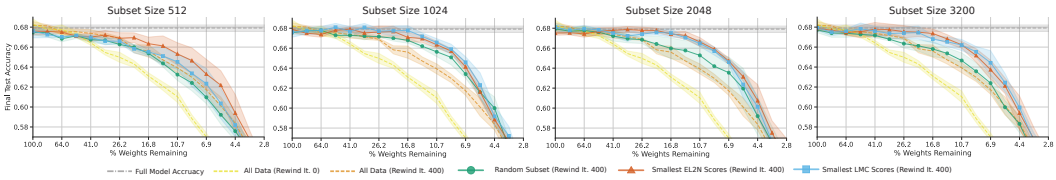


(a) CIFAR-100, ResNet-32, Batch Size 128.

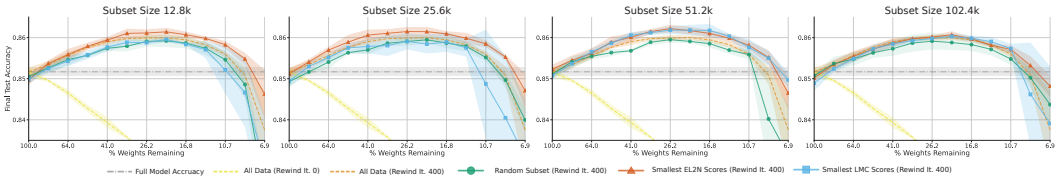


(b) CINIC-10, ResNet-56, Batch Size 256.

Figure 10: Scatter plot of the relationship between the EL2N score computed at 1600 iterations and the Linear Mode Connectivity (LMC) score computed at iteration 400, 800, and 1600 iterations.



(a) CIFAR-100, ResNet-32, $t^*/2 = 400$ LMC scores computed after 1600 iterations (batch size 128).



(b) CINIC-10, ResNet-56, $t^*/2 = 400$. LMC scores computed after 800 iterations (batch size 256).

Figure 11: Performance of IMP pre-training using data subsets determined by smallest LMC scores (solid light blue curve with squares). This is compared against random subsets (solid green curve with circles) and easiest examples (solid red curve with triangles). Pre-training is performed with learning rate 0.4 and batch size 128 for CIFAR-100 and learning rate 0.1 and batch size 256 for CINIC-10. Several of these runs are included in the scatters plot in the right column of Figure 5 with the method labelled as “Lowest Train Loss Barrier."

In our experiments, we computed the LMC score by averaging over three children runs from two replicates, so that each replicate provided three train loss barriers (one for each pair of children) and the total average was taken over six instances.

We visualize the correlation between EL2N scores and LMC scores. The scatter plots in Figure Figure 10 show that low EL2N score examples tend to be the ones for which the LMC score goes

down rapidly with spawning time, and is nearly-zero early in training. However, the opposite is not true, and high EL2N scores have a wide range of per-example loss barriers.

D.3 Pre-training on Low LMC Scores

Figure 11 shows that rewinding points obtained by training on low LMC score examples dominate random example pre-training, but are (weakly) dominated by training on low EL2N score examples. We thus conclude that low EL2N examples are more representative of the necessary and sufficient subset of examples needed to find a good rewinding point.

E Full Results

E.1 Comparison to Forgetting Scores

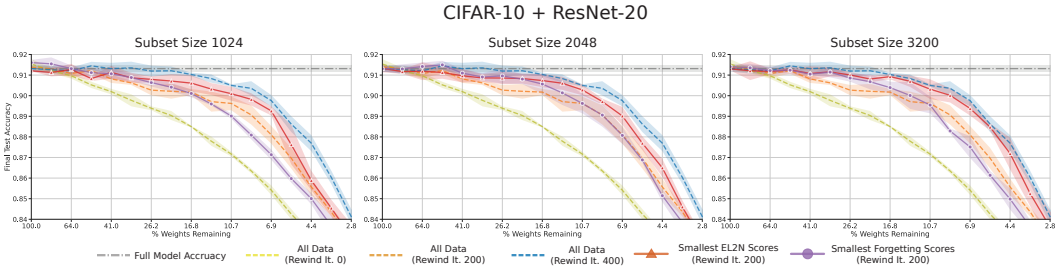


Figure 12: Comparison of pre-training on smallest EL2N scores and forgetting scores on CIFAR-10 + ResNet-20. On this dataset, we observe superior performance of EL2N scores.

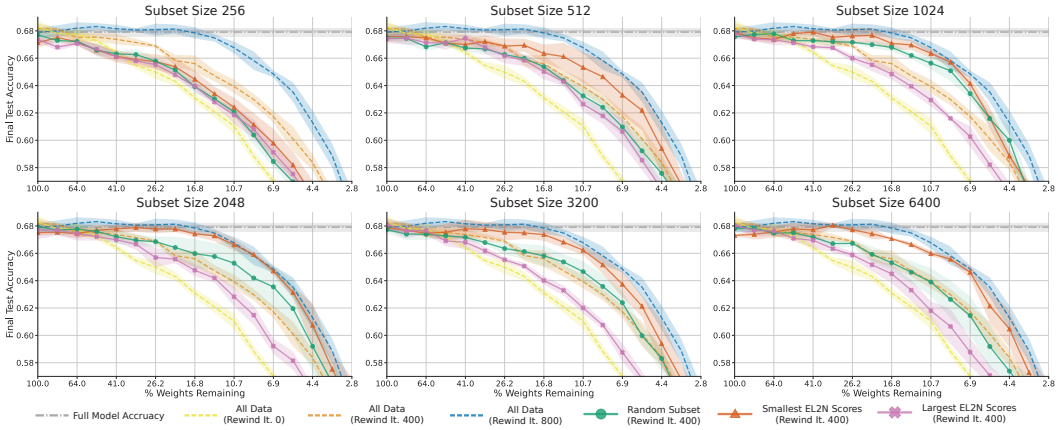
In Figure 12, we compare pre-training on the smallest EL2N scores vs. smallest forgetting scores. The forgetting score of an example is defined by Toneva et al. [24] as the number of times the example is included in a mini-batch and the network switches from a correct to incorrect classification of that example—a "forgetting event". As seen in the results, pre-training on the subset of smallest forgetting scores underperforms using the subset of smallest EL2N scores computed early in training for subset sizes 1024, 2048, and 3200.

There are two reasons why we might expect forgetting scores to underperform EL2N scores. First, forgetting scores as defined here look at example difficulty over the whole course of training whereas we computed the EL2N scores early in training at 16 epochs. For IMP pre-training, we are most interested in finding examples which the network finds easy early in training. Second, for many of the easy examples the forgetting score is 0 meaning the example is learned once and never forgotten. If this is true for a large number of examples, this ranking is less useful when trying to choose a small subset size (for both scores, the subset was class-balanced).

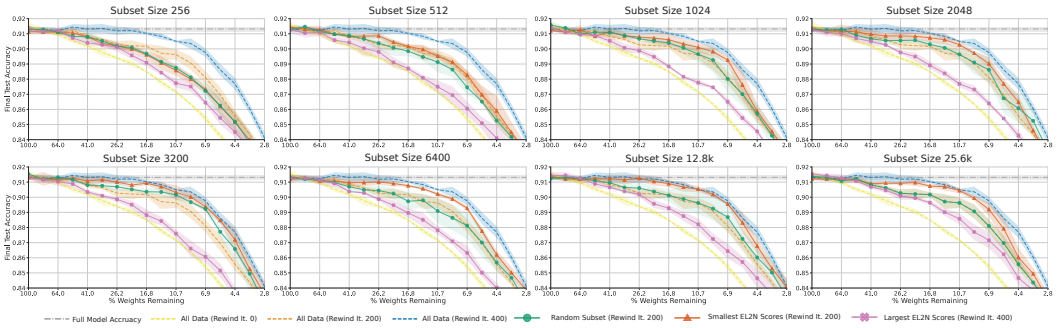
Our goal was to identify training data subsets that suffice for pre-training. Our choice of metric was thus intentional: EL2N scores capture how difficult it is to learn an example early in training, which intuitively correlates to stability of training and error barriers.

E.2 Results Across Subset Size

Here we present the full set of experiments performed for the results in the main text, sweeping across subset size. Figure 13 and Figure 14 show the results for performing IMP pre-training with different data subsets across a range of different subset sizes. Figure 15 shows the result for performing IMP pre-training with the dataset corrupted by randomized label noise (the original dataset is then used for the mask search and sparse training phase).

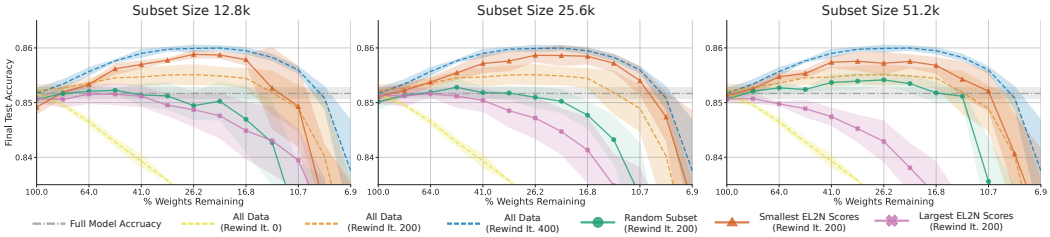


(a) CIFAR-100, ResNet-32, $t^* = 800$, $t^*/2 = 400$.

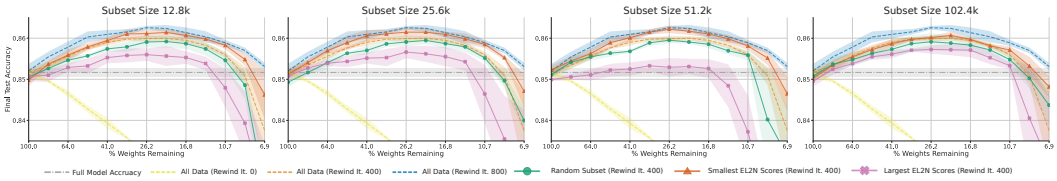


(b) CIFAR-10, ResNet-20, $t^* = 400$, $t^*/2 = 200$.

Figure 13: Extended results for Figure 2 and Figure 3 across all subset sizes considered on CIFAR-10 and CIFAR-100. For each dataset, pretraining was performed with all data for $t^*/2$ steps (dashed orange curve) and t^* steps (dashed blue curve). Performing IMP as an initialization is included as a baseline (dashed yellow curve). IMP pre-training was then performed with three different data subsets for $t^*/2$ steps: random examples (solid green curve with circles), easiest examples (solid red curve with triangles), and hardest examples (solid pink curve with crosses). For both CIFAR-100 and CIFAR-10, a learning rate of 0.4 and batch size 128 was used during the pre-training period.



(a) CINIC-10, ResNet-56, $t^* = 400$, $t^*/2 = 200$.



(b) CINIC-10, ResNet-56, $t^* = 800$, $t^*/2 = 400$.

Figure 14: Extended results for Figure 2 and Figure 3 across all subset sizes considered on CINIC-10. For each dataset, pretraining was performed with all data for $t^*/2$ steps (dashed orange curve) and t^* steps (dashed blue curve). Performing IMP as an initialization is included as a baseline (dashed yellow curve). IMP pre-training was then performed with three different data subsets for $t^*/2$ steps: random examples (solid green curve with circles), easiest examples (solid red curve with triangles), and hardest examples (solid pink curve with crosses). For CINIC-10, a learning rate of 0.1 and batch size 256 was used during the pre-training period.

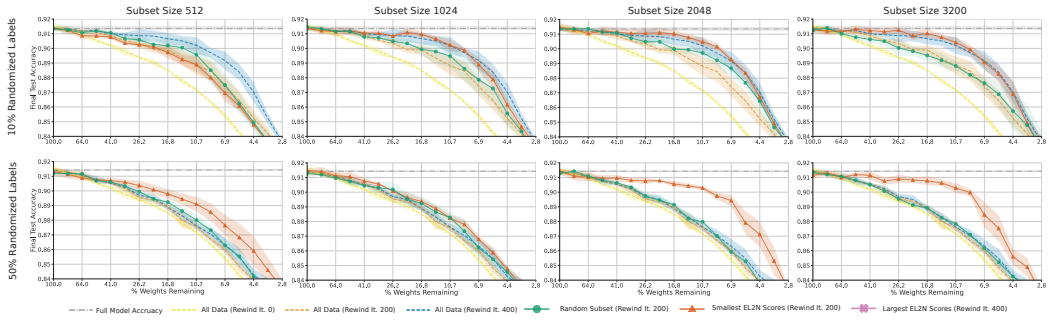


Figure 15: Extended results for Figure 4. In the top row, pre-training is performed on the dataset with 10% randomly corrupted labels; in the bottom row, pre-training is performed with 50% randomly corrupted labels. The corruption is performed once and then is held the same across subset sizes and replicates. Here $t^* = 400$, and the random and easy data subsets are trained for $t^*/2 = 200$ steps. Pre-training was performed with a learning rate of 0.4 and batch size of 128.