Pre-Training on a Data Diet: Identifying Sufficient Examples for Early Training[†]

Mansheej Paul [*1] **Brett W. Larsen** [*1] **Surya Ganguli** [1 2] **Jonathan Frankle** [3 4 5] **Gintare Karolina Dziugaite** [6 7]

Abstract

A striking observation about iterative magnitude pruning (IMP; Frankle et al. 2020a) 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 through the lens of the data distribution. 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. Combined, these results provide new insight into the role played by data in the early phase of training.

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 (Frankle & Carbin, 2018; Frankle et al., 2020a) 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; Frankle & Carbin (2018); Frankle et al. (2020a)), Remarkably, even after all these rounds of training, we do not find trainable sparse sub-networks if we rewind to the random initialization; thus, the first few hundred steps of dense network training are 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.

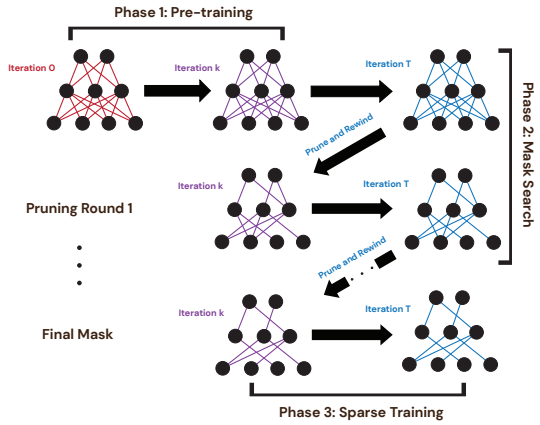


Figure 1. Three phases of iterative magnitude pruning (IMP) with weight rewinding (Frankle et al., 2020a). A dense network is trained in the pre-training phase for t_r iterations, where t_r is referred to as the rewinding iteration, and $\mathbf{w}_{t_r}$ is the state of the network at the rewinding point. The mask search phase produces a sparse sub-network 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 sub-network to convergence, starting with weights $\mathbf{w}_{t_r}$.

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

[*]Equal contribution [†]Full paper: https://arxiv.org/abs/2206.01278 [1]Stanford [2]Meta AI [3]MIT [4]Mosaic ML [5]Harvard [6]Google Brain [7]Mila; McGill. Correspondence to: Mansheej Paul <mansheej@stanford.edu>, Brett W. Larsen <bwlarsen@stanford.edu>, Gintare Karolina Dziugaite <gkdz@google.com>.

First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022, Baltimore, Maryland, USA, PMLR 162, 2022.

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.

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 $\mathbf{w}_{t_r}$
4: **for** $n \in \{1, \ldots, N\}$ **do**
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: **end for**
8: Train the final network $\mathbf{m} \odot \mathbf{w}_{t_r}$. Measure its accuracy.

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 sub-networks. The *lottery ticket hypothesis* (Frankle & Carbin, 2018) states that any standard neural network "contains [at initialization] a sub-network 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 sub-networks* (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 (Frankle et al., 2020a; Yu et al., 2020; Chen et al., 2020; Vischer et al., 2021; Kalibhat et al., 2020).

Empirical evidence for this phenomenon comes via a procedure that finds such sub-networks retroactively after training the entire network. This procedure, called *Iterative Magnitude Pruning* (IMP, Figure 1; Frankle et al., 2020a) is based on standard iterative pruning procedures (Han et al., 2015) and is outlined in Algorithm 1.

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 sub-networks. 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. (2020b) showing that the standard pre-training phase could be replaced by a much longer self-supervised phase.

There are many other ways to obtain pruned neural networks (e.g., Janowsky, 1989; LeCun et al., 1990; Han et al., 2015; Zhu & Gupta, 2017; Evci et al., 2020). 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.

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. (2021). 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 $\mathbf{w}_t$ conditioned on the training data.

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 subset at random while preserving class balance, or based on the EL2N scores.

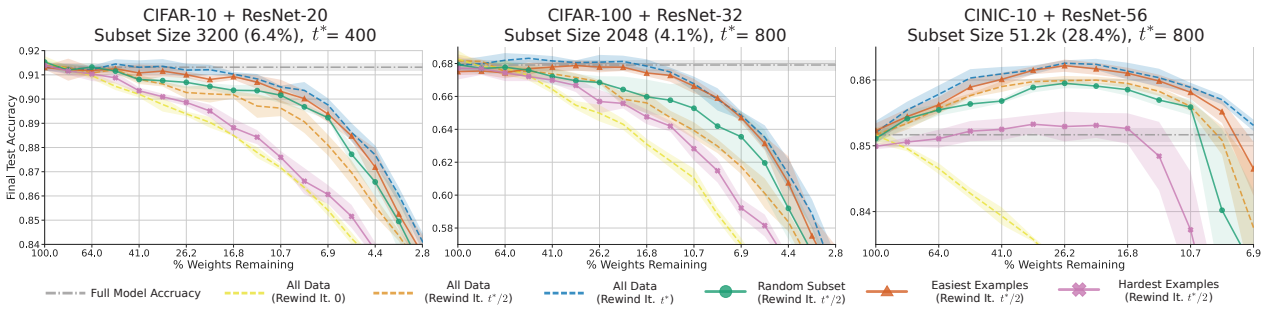


Figure 2. For a given rewind step $t_r = t^*/2$, training on a small fraction of random data during the pre-training phase of IMP leads to matching initializations (compare the solid green with circles and dashed orange curves) across dataset, network, and hyperparameter configurations. Using just the 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 4 and Appendix B.

3. Training Data and IMP 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.

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 $\mathbf{w}_0$. 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 $\mathbf{w}_{t^*}$.

We empirically find that certain surprisingly small subsets S and rewind step $t_r < t^*$ lead to matchining initializations fore 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 (Paul et al., 2021).

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 4 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

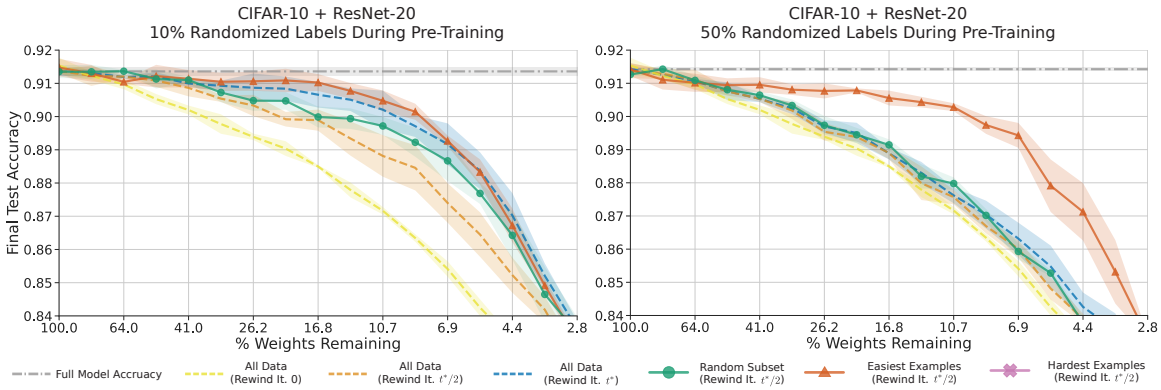


Figure 3. Pre-training on a random subset or all data is not robust to label noise during this initial phase of IMP. However, pre-training with the easiest data as scored by EL2N scores computed from the corrupted dataset is robust. In both the left (10% randomized labels) and right (50% randomized labels), 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 B.

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^*.

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 barely 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 (Paul et al., 2021). 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 all data or a random subset is not robust to corruption with random label noise (Zhang et al., 2016) during the pre-training phase. As seen in Figure 3, 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 3). This is because examples with randomized labels are hard (Paul et al., 2021) 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, in this case the easiest examples as defined by the lowest EL2N scores.

4. Discussion

Recent empirical evidence has shown that deep neural network optimization proceeds in several distinct phases of training (Frankle et al., 2020b; Fort et al., 2020). 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 to enable a more efficient scheme where the early part of training focuses on a small fraction of the overall training set while the remaining data is loaded in. Though this work does not provide an improved algorithm for obtaining sparse networks, we believe our results provide 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

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

Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., and Carbin, M. The lottery ticket hypothesis for pre-trained bert networks. 2020.

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

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

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S. 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.

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

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

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

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

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

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

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

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

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in neural information processing systems*, pp. 598–605, 1990.

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

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

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

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

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

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 (Krizhevsky et al., 2014), and CINIC-10 (Darlow et al., 2018) 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 (He et al., 2016). 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 were computed by training on the corrupted dataset. As seen in Figure 5 of (Baldock et al., 2021), 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 (Paul et al., 2021). Thus, the noisy labels are filtered out by pre-training on the examples with the lowest EL2N scores.

Hyperparameters. Networks were trained with stochastic gradient descent (SGD). t^* was chosen for each dataset such that $t_r = t^*$ produces a matching initialization when trained with all data; 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%

EL2N score computation To calculate EL2N scores for a dataset, we follow the process outlined in (Paul et al., 2021). 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.

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.

B. Full Results

Here we present the full set of experiments performed for the results in the main text. Figure 4, Figure 5, and Figure 6 show the results for performing IMP pre-training with different data subsets across a range of different subset sizes. Figure 7 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).

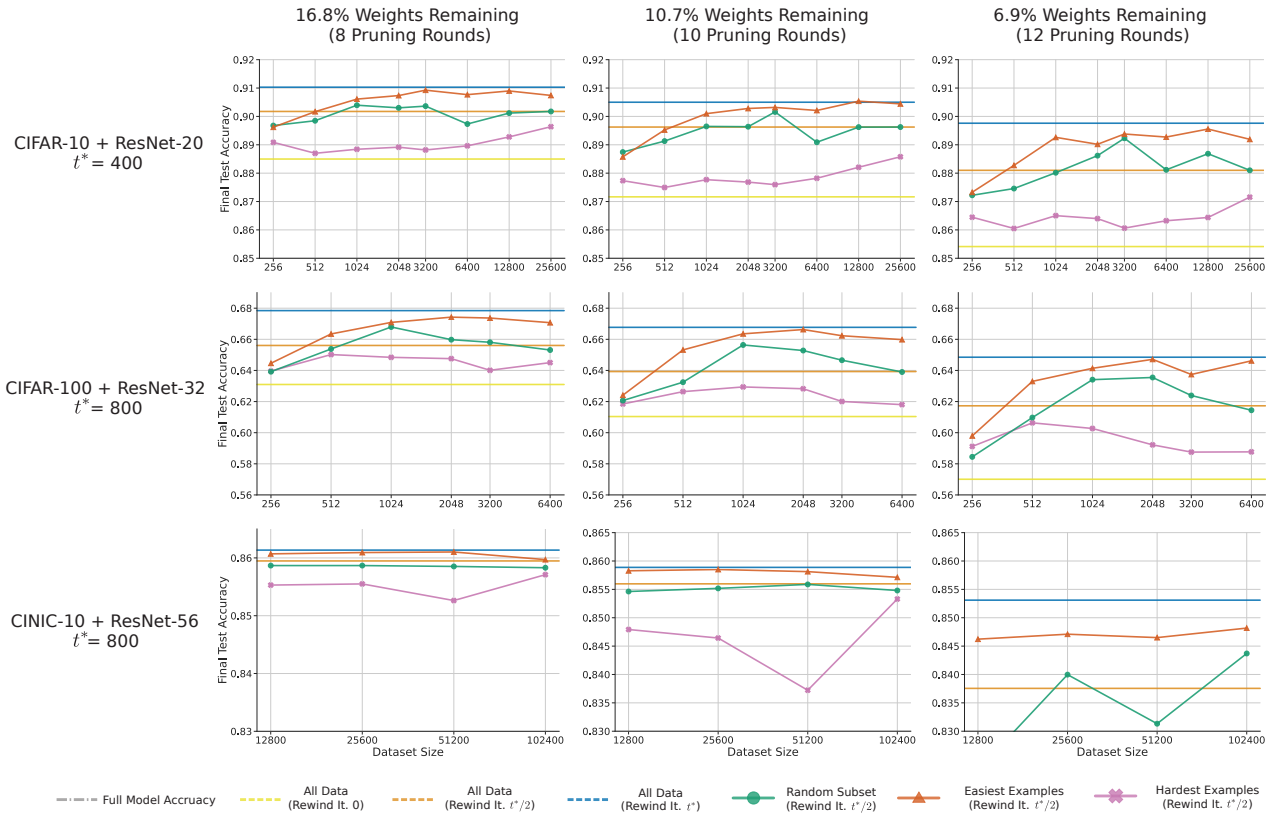
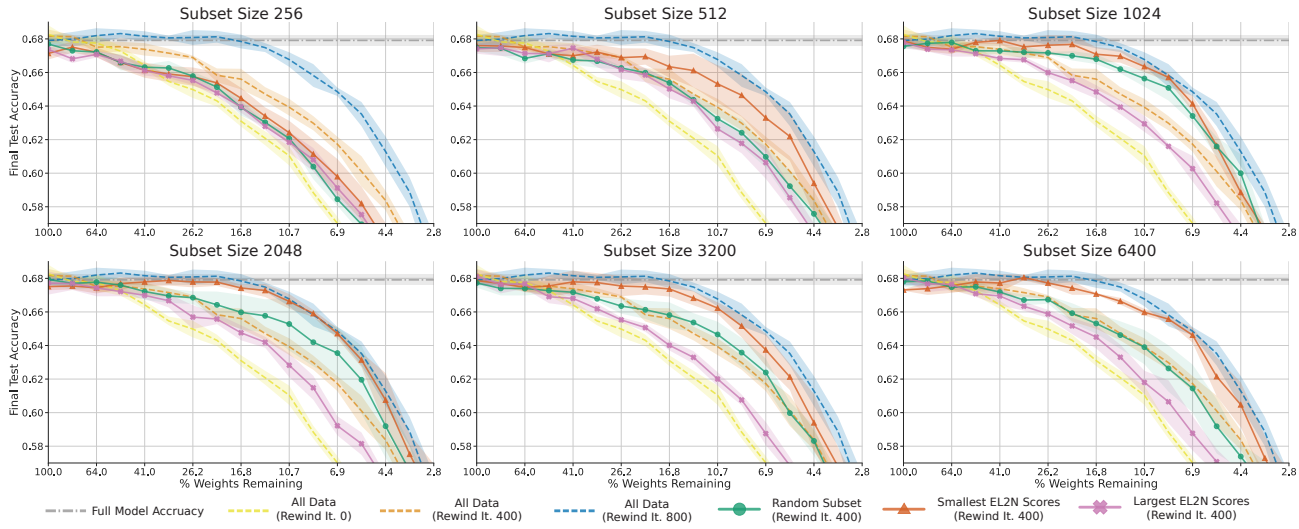
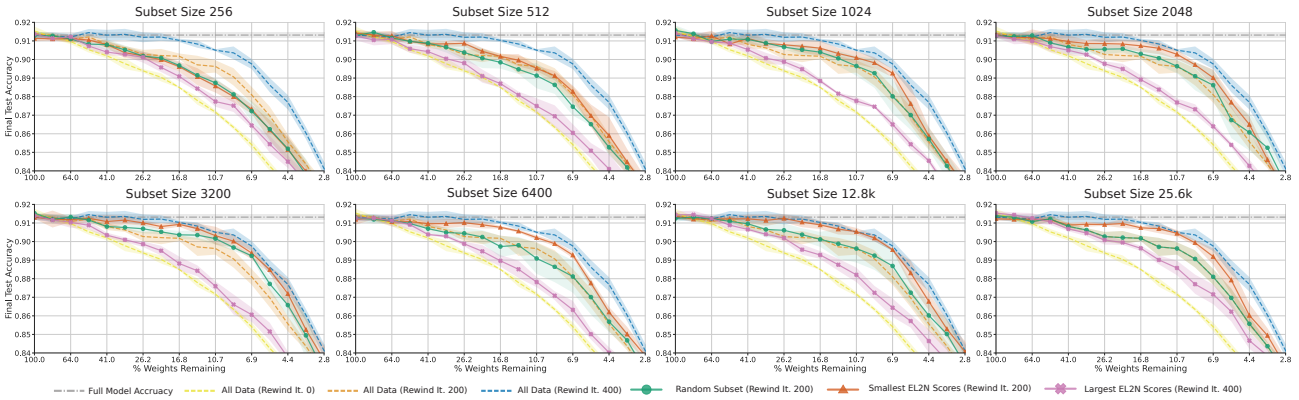


Figure 4. A summary of the the dependence on subset size for the style of experiments described in Figure 2. The first column represents the performance across subset size for the fixed sparsity 16.8% weights remaining or 8 rounds of pruning. The subsequent columns show the same for 10.7% (10 pruning 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.

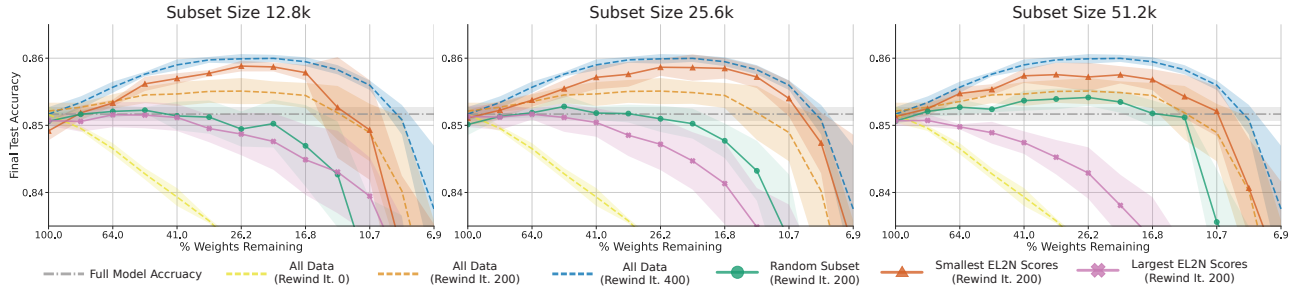


(a) CIFAR-100, ResNet-32, $t^* = 800$, $t^*/2 = 400$.

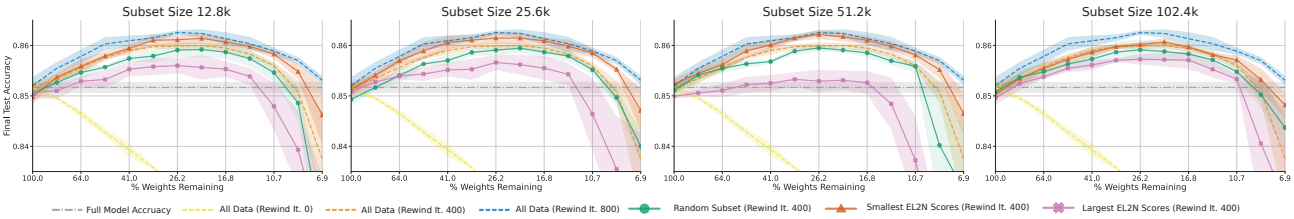


(b) CIFAR-10, ResNet-20, $t^* = 400$, $t^*/2 = 200$.

Figure 5. Extended results for Figure 2 and Figure 4 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 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 6. Extended results for Figure 2 and Figure 4 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 256 was used during the pre-training period.

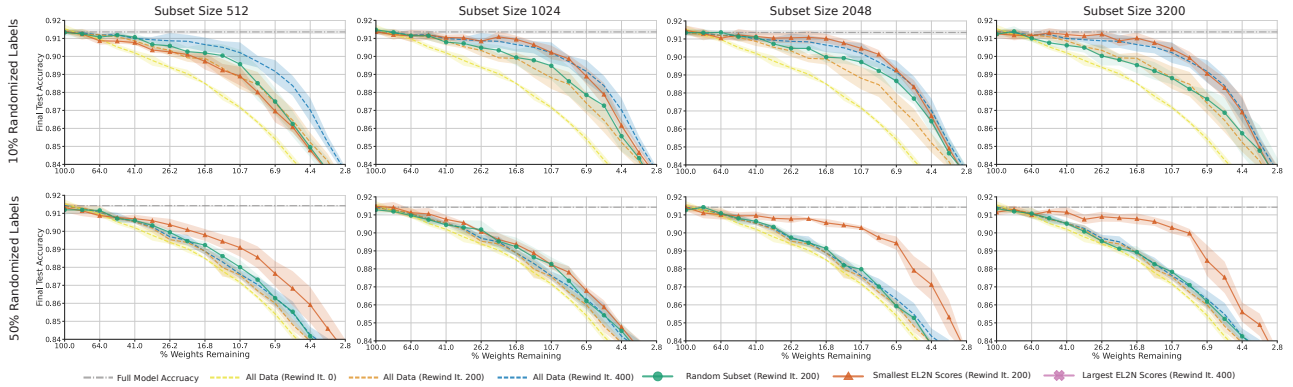


Figure 7. Extended results for Figure 3. In the top row, pre-training is performed on the dataset with 10% randomly corrupted labels; in the bottom row, pre-training is performed with 50% randomly corrupted labels. The corruption is performed once and then is held the same across subset sizes and replicates. 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.