# Lowering the Pre-training Tax for Gradient-based Subset Training: A Lightweight Distributed Pre-Training Toolkit

Yeonju Ro [1] Zhangyang (Atlas) Wang [1] Vijay Chidambaram [1] Aditya Akella [1]

## Abstract

Training data and model sizes are increasing exponentially. One way to reduce training time and resources is to train with a carefully selected subset of the full dataset. Prior work uses the gradient signals obtained during a warm-up or "pre-training" phase over the full dataset, for determining the core subset; if the pre-training phase is too small, the gradients obtained are chaotic and unreliable. As a result, the pre-training phase itself incurs significant time/resource overhead; prior work has not gone beyond hyperparameter search to reduce pre-training time. Our work explicitly aims to reduce this **pre-training tax** in gradient-based subset training. We develop a principled, scalable approach for pre-training in a distributed setup. Our approach is *lightweight* and *minimizes communication* between distributed worker nodes. It is the first to utilize the concept of model-soup based distributed training *at initialization*. The key idea is to minimally train an ensemble of models on small, disjointed subsets of the data; we further employ data-driven sparsity and data augmentation for local worker training to boost ensemble diversity. The centralized model, obtained at the end of pre-training by merging the per-worker models, is found to offer stabilized gradient signals to select subsets, on which the model is further trained. We have validated the effectiveness of our method through extensive experiments on CIFAR-10/100, and ImageNet, using ResNet and WideResNet models. For example, our approach is shown to achieve **15.4**× pre-training speedup and **2.8**× end-to-end speedup on CIFAR10 and ResNet18 without loss of accuracy. Codes will be fully released.

[*]Equal contribution [1]The University of Texas at Austin. Correspondence to: Yeonju Ro <yro@cs.utexas.edu>.

*Proceedings of the $40^{th}$ International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. 

## 1. Introduction

In the era of deep learning, neural networks are consistently getting bigger, and so are the dataset sizes (Gao et al., 2021; Sun et al., 2020). Training with gigantic datasets demands significant computation resources such as CPUs and GPUs, and high-capacity memories are required to load and process samples. When it comes to distributed training, large datasets also induce significant communication between storage and compute nodes as well as between worker nodes (e.g., to aggregate gradients).

Subset (or coreset) training has been explored as a potential workaround to the large dataset explosion (Agarwal et al., 2005; Maalouf et al., 2019). The key idea is to train on a small, but important, subset instead of the full dataset; this reduces both training time and resource use. Researchers have proposed a variety of algorithms to select the most valuable data (e.g., based on Shapley value (Tripathi et al., 2020), geometric distance (Sener & Savarese, 2017), and submodular functions (Wei et al., 2014a; 2015; 2014b; Iyer et al., 2021)). Among these, the recently proposed gradient-based methods (Toneva et al., 2018; Paul et al., 2021; Killamsetty et al., 2021a; Mirzasoleiman et al., 2020; Killamsetty et al., 2021b) have demonstrated state-of-the-art accuracy. These approaches generally use gradient signals for data valuation. Score-based methods (Toneva et al., 2018; Paul et al., 2021) give an importance value to each data item based on its gradients and select items with high values for subset training. Gradient-matching-based methods (Killamsetty et al., 2021a; Mirzasoleiman et al., 2020) pick the data subset whose weighted gradients can best match the full training (or validation) set's gradients.

Gradient-based subset training methods run into a well-known issue: since early-stage gradients obtained from nearly untrained models are chaotic and unreliable, they can deteriorate the quality of the chosen subset. Most prior works hence introduce a default **"pre-training"** or warm-up stage and later perform subset selection using the pre-trained model gradients. The typical pre-training approach is to train a model using the *entire data* over several epochs (Section 3.1). Despite effectively boosting the accuracy, this pre-training convention is clearly *not free of cost*: it can incur significant time/resource overhead; prior work did not seek to optimize this cost beyond doing ad-hoc hyperparameter search to reduce the number of epochs in pre-training (Killamsetty et al., 2021a). There is a pressing need for work on systematically exploring how to further lower the **"pre-training tax"** in a principled, scalable, and resource-efficient manner.

### 1.1. Our Contributions

We present an efficient *end-to-end* training framework for gradient-based subset training algorithms. We explicitly optimize the pre-training tax by developing a novel **"distributed-friendly"** lightweight pre-training approach, which forms the *primary emphasis* of our paper. Our approach spreads pre-training work across multiple workers, yet with **two unique targets** to ensure we "lower the tax":

- Workers do not have to synchronize nor communicate during pre-training. Ideally, we only perform local training at each worker and then conduct a "one-shot" aggregation of results at a central worker (which will continue the main training) by the end of pre-training.
- We do not ship the full data to each worker (too heavy for both communication and per-worker local training). Ideally, we send only a tiny random subset to each worker, and the union across all workers is just a small subset of the entire data (rather than full coverage).

Correspondingly to achieve the two goals, we empower our distributed pre-training recipe with **two tailored ideas**:

- We introduce a model-soup-inspired (Wortsman et al., 2022b) efficient ensembling strategy to pre-training, that eliminates communication between distributed models before the pretraining completes. While recent works (Li et al., 2022; Wortsman et al., 2022a) already demonstrate similar ideas for data-parallel multi-node finetuning of large pre-trained models without communication, to our best knowledge, we are the first to reveal the "model-soup" idea to effectively work for distributed training *at initialization* (for our customized purpose of subset selection).
- To strengthen local worker training effectiveness over tiny random subsets, we leverage data-driven sparsity as well as aggressive data augmentation as two regularization ways to mitigate local overfitting, which also boosts the ensemble diversity.

After the distributed pre-training is completed, the centralized worker collects and aggregates all local models into one pre-trained model, to then generate gradients used for selecting the subset (which can be done by any off-the-shelf algorithm). After that, the main training continues.

Our result shows that the proposed framework reduces the variance of subset training using extremely low time and data costs, as well as improving the final accuracy. We have validated the effectiveness of our method through extensive experiments on CIFAR-10/100, and ImageNet, using ResNet and WideResNet models. For example, it reduces the pre-training time by up to 15.4× compared to the baseline subset selection algorithm and 2.8× compared to the full dataset baseline without compromising accuracy in ResNet18 and CIFAR10.

## 2. Background

### 2.1. Subset selection algorithms

Coreset is a promising approach that reduces computation and memory cost during training (Mirzasoleiman et al., 2020). One representative approach is using Shapley value (Tripathi et al., 2020), which represents the marginal contribution of a sample measured by leaving the sample out of the original set. This approach is intuitive, yet hard to measure in large-scale data. Other approaches use the geometric distance between data items, or submodular greedy approximation (Sener & Savarese, 2017; Sinha et al., 2020).

Recent works tend to use error or loss during the training as their metric for valuation. Forgetting events (Toneva et al., 2018) counts how many times each data item is misclassified or forgotten during training. The authors concluded that unforgettable examples can be removed from the training set without hurting generalization. GraNd score measures the expected loss of gradient norm in early epochs of training, approximated by the error $\ell_2$ norm (Paul et al., 2021).

More directly, a subset can be selected to minimize the difference between gradients from the subset and the gradients from the original dataset (Mirzasoleiman et al., 2020). Gradient-matching can happen regularly during the training as well (Killamsetty et al., 2021a) to dynamically update importance samples in the subset. However, to infer a reliable importance value of each data item, gradients need to be stabilized before the valuation - that often cannot be met for the early-stage gradients from severely under-trained weights. Most prior works hence introduce a default "pre-training" or warm-up stage on the full dataset; then perform subset selection using pre-trained model gradients.

Despite effectively boosting the final accuracy, the pre-training incurs non-negligible, even significant time /resource overhead. For example, the pre-training of recent works can take 15∼40 epochs (Guo et al., 2022; Killamsetty et al., 2021a; Toneva et al., 2018). To our best knowledge, limited work explicitly focused on optimizing this "pre-training tax" issue. GradMatch (Killamsetty et al., 2021a) did an ad-hoc hyper-parameter search to determine the epoch number allocation between pre-training and main training, but their objective was to boost the test accuracy instead of reducing the pre-training or end-to-end time cost.

### 2.2. Model ensembling for large pre-trained models

As large-scale pre-trained models became prevalent, researchers put efforts into fine-tuning the pre-trained model with different hyper-parameters to achieve the best accuracy for downstream tasks (Kolesnikov et al., 2020; Girshick et al., 2014). As a common fine-tuning process, each pre-trained model is optimized with different hyper-parameters separately, and one model that performs the best is picked. However, more recent work showed that instead of discarding the rest, an ensemble of fine-tuned models trained with different hyper-parameters outperforms a single model's accuracy and improves model robustness because of increased diversity in the ensemble (Wang et al., 2020).

One issue of conventional ensembling is that it requires additional processing during inference (Breiman, 1996; 2001). To resolve this issue, recent work explore merging multiple models by *weight averaging*. The notable idea of "Model Soup" (Wortsman et al., 2022b) merged models fined-tuned from the same pre-trained model for the same downstream task, but with different hyper-parameters, without incurring any additional inference or memory costs at inference. Their obtained model provides significant improvements over the best model in a hyperparameter sweep on ImageNet. In a similar vein, model recycling (Ramé et al., 2022) merged the weights of fine-tuned models optimized for diverse auxiliary tasks. This simple technique improved both in-distribution and out-of-distribution test accuracy without increasing inference latency. BTM and lo-fi (Li et al., 2022; Wortsman et al., 2022a) extended the models soup idea to a distributed fine-tuning setting, merging models fine-tuned independently with *different data distributions* without any communication during the training.

Note, though, that all "model soup"-type ideas explored so far (Wortsman et al., 2022b; Li et al., 2022; Wortsman et al., 2022a; Ramé et al., 2022) are focused on the fine-tuning setting (one or multiple downstream tasks; centralized or distributed) from a large pre-trained model. Meanwhile, applying the same idea to training from scratch looks quite risky if not daunting, since there is no guarantee that models trained independently from random initialization would organically stay in the same solution basin (Ainsworth et al., 2022) or linearly connected mode (Frankle et al., 2020) (a commonly believed prerequisite for model merging).

Yet, we empirically discover in this paper, that the model soup idea at least works well in a constrained context for "training from scratch": if our goal is not to obtain a highly performant model directly, but rather a model whose gradients are reliable enough for indicating the data sample importance. We leverage this finding to pre-train efficiently in a distributed setup, and the weight-averaged model provides consistent and robust gradients for data valuation. To our best knowledge, our work is the first to effectively exploit the "model-soup" idea in pre-training from scratch.

## 3. Method

### 3.1. Overview of the Framework

We propose a lightweight and scalable end-to-end training framework that provides robust initialization for gradient-based subset algorithms at low cost. Figure 1 overviews our subset training framework which is divided into three stages: pre-training, subset selection, and main training.

The objective of our framework is to reduce the end-to-end cost. End-to-end cost consists of 1) pre-training cost, 2) subset selection cost, and 3) main training cost. Pre-training cost is the time taken in training an initial model to get the initial gradient. Subset selection cost is the time taken in running a subset algorithm. Subset selection time is a one-time cost and varies by different algorithms. The main training time is the time taken in training a model with the selected subset. Our work mainly focuses on reducing pre-training costs, or the "pretraining tax", and the main training time, with minimal loss of accuracy.

In our framework, the subset training pipeline starts from scratch. For a given dataset, we could train a random initial model with less than three epochs to get an initial model (the "initialization" stage). In prior work, the pre-trained phase takes 15–40 epochs even for small datasets (e.g., CIFAR10) (Guo et al., 2022; Killamsetty et al., 2021a; Toneva et al., 2018). However, this corresponds to **20–40% of the entire training time** considering 180–200 epochs of main training is common in CIFAR10/CIFAR100 and 90–100 epochs is common in ImageNet-1K (He et al., 2016).

A negative consequence of naively reducing pre-training epochs is the deteriorated quality of gradients for the subset selection. We hence refer to **distributed training** for accelerating and potentially scaling up pre-training for subset selection, which was unexplored by prior arts yet: $n$ different workers train in parallel from the same initial model $\theta_*$ and then ensemble their trained models, which shall expect to gain us a more stable pre-trained model compared to a single worker using the same amount of clock time. However, distributed training inherently has challenges of **increased latency** due to communication and asynchronicity. In the following sections, we will discuss the detailed challenges and mitigations to resolve the issues. Each distributed worker performs local training with a random subset (Section 3.3) for $K$ steps, regularized by data-driven sparsity as well as random augmentations. Then, locally trained models ($\theta_0 \sim \theta_{n-1}$) are aggregated with model-soup based ensembling (Section 3.2) to get a pre-trained model $\theta_p$.

The framework then runs a subset selection algorithm using the gradients from the pre-trained model $\theta_p$. We can use any

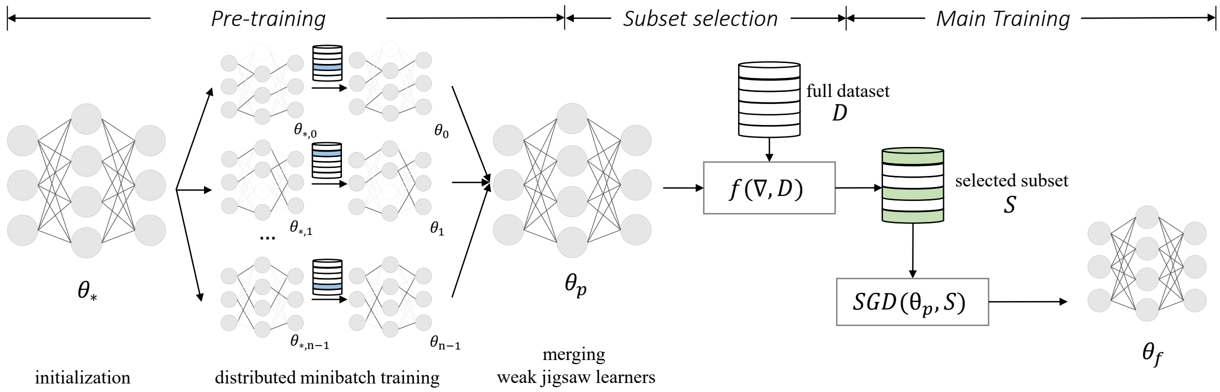


*Figure 1.* **Our Proposed Subset Training Framework Overview**. Pre-training starts from the initial model $\theta_*$, which is distributed to different workers ($\theta_{*,0} \sim \theta_{*,n-1}$) as the common initialization. Each worker weakly trains it with a different small subset from the full dataset, and leverages model sparsification as well as data augmentation to increase model diversity while reducing overfitting. The trained local models ($\theta_0 \sim \theta_{n-1}$), termed *jigsaw weak learners*, will be merged into a dense model in the end, by averaging their weights. With the merged pre-trained model $\theta_p$ to supply gradients, the subset selection algorithm can run and return a selected subset $S$ from the full dataset $D$. Using the subset $S$ and the pre-trained model $\theta_p$, we continue the main training stage to obtain the final model $\theta_f$.

subset selection algorithm that is based on gradients or task loss. Particularly, many gradient-based algorithms (Mirzasoleiman et al., 2020; Killamsetty et al., 2021b;a) perform re-selection during the main training to further improve the accuracy even with a lower fraction of data. With reselection, the most recently updated gradient during the main training can be used for periodically updating the subset. In our experiment, we also follow the reselection mechanism to update the subset every 20 epochs, and we find it to boost our performance too.

### 3.2. Model-Soup Inspired Efficient Ensembling

One of the main challenges of distributed training is the high communication cost of sending a large dataset. This adds significant latency in the training pipeline and becomes a major bottleneck. Communication of parameters between works may also cause synchronization problems.

To tackle these challenges and make the initial training scalable, we set two regimes. First, we do not synchronize nor communicate between workers during the training but only allow local training of workers. Second, we do not ship the entire dataset to workers; we select a small, random subset, break it into disjoint parts, and send one part to each worker. We kill those two birds with the same stone: an efficient ensemble initialization inspired by recent work (Wortsman et al., 2022b;a; Ramé et al., 2022).

Following (Wortsman et al., 2022b), we propose to create $n$ instances of the same $\theta_*$ at the central worker, and send each to $n$ distributed workers, which train them independently in isolation. Different from the original model soup (Wortsman et al., 2022b), we do not vary the hyperparameter configuration across workers, but instead leverage the natural randomness perturbations by data dispatching. We send only a small random subset to each worker instead of sending the full dataset: i.e., during the local training, each local model ($\theta_{*,0}\sim\theta_{*,n-1}$) is trained with its own possessed $M$ (hyperparameter of choice) minibatches of data; each worker iterates over its minibatches for $K$ steps (another hyperparameter of choice). We maintain the assigned minibatches to not overlap across different models, and the amount of the entire data sent out across all constitutes only 2∼25% of the full dataset. That serves dual purposes: (i) create more diverse local models that will robustify the model soup averaging, and (ii) avoid the high communication costs and latency incurred by moving larger subsets or the full dataset.

The local training will be performed without any communication between workers, similarly to (Wortsman et al., 2022a). Its main challenges lie in how to effectively train from scratch with very few-shot data per worker - which we will address in the next section. Once all local models are trained and sent back to the central worker, the aggregation follows the greedy interpolation soup procedure as described in (Wortsman et al., 2022b). Namely, we sequentially add each local model as a potential "ingredient in the soup", and only keep the model in the soup if it leads to improving performance on the validation set. For merging two ingredients, we perform a search for an optimal interpolation ratio $\alpha \in [0, 1]$ that helps in performance gain, or else the ingredient is discarded. Eventually, the outcome of the pre-training stage is a model to supply stabilized gradients.

### 3.3. Local Training with More Robustness and Diversity

Sending only a small subset of data to each worker causes a data-scarce environment that is prone to casing model overfitting. We apply two additional techniques during the local training: data-driven sparsification as a model regularization, and more aggressive random data augmentations. They also serve dual purposes: to combat the overfitting at local training, and to increase diversity across local models.

**Data-driven model sparsity**. We first apply data-driven sparsity as a regularizer to reduce overfitting while increasing model heterogeneity. In this work, we adopt the gradual magnitude pruning (GMP) (Han et al., 2015) approach, due to its simplicity and low overhead. We keep removing the smallest-magnitude weights until reaching a pre-set target sparsity rato. Instead of one uniform ratio for all workers, we further set a random sparsity ratio per worker to add another level of local diversity. In our experiments, we keep target sparsity mild for all workers, which would be drawn from a uniform distribution between [0.85, 0.95] (this means the percentage of remaining nonzero weights). We empirically find heavier sparsity to start incurring too noisy gradients, especially after strong data augmentations.

Note that the main purpose of sparsity is **not** model compression or acceleration here - although we slightly benefit from that, as a side effect, in lowering the communication load when we transmit the trained local model back to the central worker. The main role that sparsity plays here is to reduce overfitting, as recent works (Varma et al.; Chen et al., 2021) demonstrate it to a powerful regularizer in few-shot learning or generation. Another bonus here of using data-driven sparsity is that local models will form different sparse masks due to training over different, non-overlapped data subsets, further increasing the diversity of sparse models in the ensemble. We name each local model as *weak jigsaw learner* since we are merging models of different sparse architectures, and trained weakly on small and non-overlapped subsets, to obtain a single dense pre-trained model.

**Stronger random data augmentation**. Data augmentation is another common effective technique to combat overfitting. Different from hand-crafted augmentations, RandAugment (Cubuk et al., 2020) effectively searches for an optimal augmentation tailored for different models and datasets. We adopt RandAugment (e.g., its searched augmentation policy on CIFAR-10/100, ImageNet, etc.) to our local worker training and observe it to bring ever larger gains when the local subset becomes smaller (down to only 1$\sim$10 mini-batches). It effectively enriches the local dataset without communicating more real data, and also adds to worker randomness/diversity.

Two key hyperparameters matter for RandAugment: the number of augmentation types used, and how strong magnitude each augmentation takes. Through experiments, we find that **stronger data augmentations** would benefit this situation more: our final RandAugment picked random augmentation from 14 different policies of strength 9: those are clearly heavier than and replacing the default augmentations (such as random crop or random horizontal flip).

## 4. Experiment Results

In this section, we evaluate the effectiveness of our subset training framework. We aim to answer four questions:

1. Does the framework help reduce the end-to-end training time required for training?
2. How does our framework compare to prior work when training with small fractions of the dataset?
3. How does each technique used in the framework contribute to the total performance?
4. How do the hyper-parameters affect the results?

We employ three widely-used models, ResNet-18 (He et al., 2016), Wide-ResNet-28-10, and Wide-ResNet-50-2 (Zagoruyko & Komodakis, 2016), and three datasets, CIFAR10/100 (Krizhevsky et al., 2009), and ImageNet (Russakovsky et al., 2015) in our evaluation. Further details about the results can be found in the appendix.

### 4.1. End-to-end training time

| Approach | Pre-training (s) | Main training (s) | Total (s) |
|---|---|---|---|
| Full set | 0 | 2,476 | 2,476 |
| GraNd | 346 | 523 | 869 |
| This work | 22 | 284 | 306 |

*Table 1.* End-to-end training times for different training approaches. Our framework reduces total time for training by 87% compared to training on the full dataset, and by 65% compared to prior work on subset training.

Table 4.1 compares three approaches to training ResNet18 on the CIFAR10 dataset to state-of-the-art accuracy. The first is the traditional approach, training without any subset selection on the full dataset. The second is using the GraNd (Paul et al., 2021) subset selection algorithm. Training uses 20% of the dataset. Finally, our framework also runs the GraNd algorithm but uses only 10% of the dataset to train to accuracy. Despite using the same subset algorithm, our framework results in **2.8**$\times$ speedup in end-to-end training time, with **15**$\times$ reduction in pre-training time. Compared to training on the full dataset, our framework reduces end-to-end training time by **87%**, while reaching the same accuracy. These results demonstrate the efficacy of the techniques used in our framework.

### 4.2. Accuracy with small data

Next, we evaluate how our framework performs when training with small fractions of data. For this experiment, we evaluate different subset-based approaches when they train on the same amount of data, and evaluate the Top-1 accuracy (%) obtained.

| Data Fraction | Glister | This work | Improvement |
|---|---|---|---|
| 1% | 27.04±1.3 | 47.50±1.7 | +20.45% |
| 5% | 51.64±2.7 | 59.30±1.9 | +7.66% |
| 10% | 62.75±2.6 | 67.74±1.8 | +4.99% |
| 20% | 64.58±4.6 | 75.65±1.9 | +11.07% |

*Table 2.* Accuracy obtained when training with small fraction of the dataset. Our framework consistently obtains higher accuracy.

Table 4.2 shows the result of this experiment. We vary the fraction of the dataset used in training from 1% to 20%. We compare our framework with the Glister subset training algorithm (Killamsetty et al., 2021b). We choose Glister because we observed that it used the least amount of data compared to other subset algorithms.

We observe that our framework always achieves higher accuracy than Glister, even when training on small fractions of the subset. This is due to the use of data augmentation and model merging in our pre-training stage, that yield robust gradients for subset selection.

**Reduction in the variance of accuracy**. We also observed that Glister results in higher variance among top1-accuracy results than our framework. The variance in Glister's results changes with the subset size while remaining constant for our framework. We can attribute this to the robust gradients that result from our framework's pre-training stage. The improvement from our framework is consistent over different models and subset combinations.

**Summary**. Our framework achieves SOTA accuracy with significantly lower end-to-end training time. Compared to prior work on subset selection, our framework reduces the pre-training time significantly. When training on small amounts of data, our framework results in better accuracy. Our framework also provides more stable results.

### 4.3. Overall Accuracy-I/O Cost

Figure 2 shows the Top-1 accuracy (%) measured with/without our framework in different model and dataset combinations over the amount of data used. We used Glister (Killamsetty et al., 2021b) for our baseline subset selection algorithm. Here, we do not apply reselection of the dataset during subset training; instead, we used the same subset selected after pre-training to measure the robustness of the initial subset.

Note that the x-axis is a proxy for end-to-end training cost that includes pre-training and main-training. By increasing subset data fraction, the cost of end-to-end training increases. The result shows that our framework effectively boosts subset training for the same cost.

Saving cost under the same data fraction brings another advantage: with the saved time (mostly in pre-training), we can train more epochs during the main training to further improve the accuracy for a given system cost.

Our experiment in Section 4.2 showed that we can dramatically improve the accuracy and reduce the variance for small fractions (<20%) of data. Figure 2 shows that the result is consistent over different models, subset fractions and dataset combinations. In particular, the variance in the lower subset fraction cases is greatly reduced (as depicted by the shaded region) – the more robust initial gradients from our approach appear to have a greater impact in this regime. In the appendix, we show additional experimental results measuring with different algorithms (e.g., GraNd (Paul et al., 2021), Gradmatch (Killamsetty et al., 2021a)). We observed that the improvement is significant over different algorithms.

### 4.4. Ablation Study

We now study the extent to which different techniques we proposed – model merging, model pruning, and data augmentation contribute to the overall accuracy. We study three combinations and compare them with over overall approach: ❶) refers to just model merging, where, after initialization (Figure 1) each distributed worker trains the same initialized model using random mini-batches and without using data augmentation. In ❶ + ❷, each worker trains a pruned model (so there is model diversity across the workers), but no data augmentation is used. In ❶ + ❸, each worker trains on the same model obtained from initialization, but mini-batches at a worker are combined with data augmentation.

Table 3 shows the Top-1 Accuracy (%) and standard deviation of the final model trained with the different combinations above as well as our final technique (❶ + ❷ + ❸). We used Glister (Killamsetty et al., 2021b) as our main subset selection algorithm and study the results for two different subset fractions (10% and 70% of the dataset).

The result shows that improvements from data augmentation and model pruning are consistent in both low and high subset fractions. Compared to just model merging (❶), accuracy can be improved and the variance can be reduced with model pruning (❶ + ❷). Moreover, data augmentation (❶ + ❸) dramatically amplifies the accuracy and stabilizes the variance. This is mainly because we only assign a small number of minibatches for each worker to control I/O cost – 1 batch for CIFAR10, 5 batches for CIFAR100, and 10 batches for ImageNet (detailed configuration for the experi-

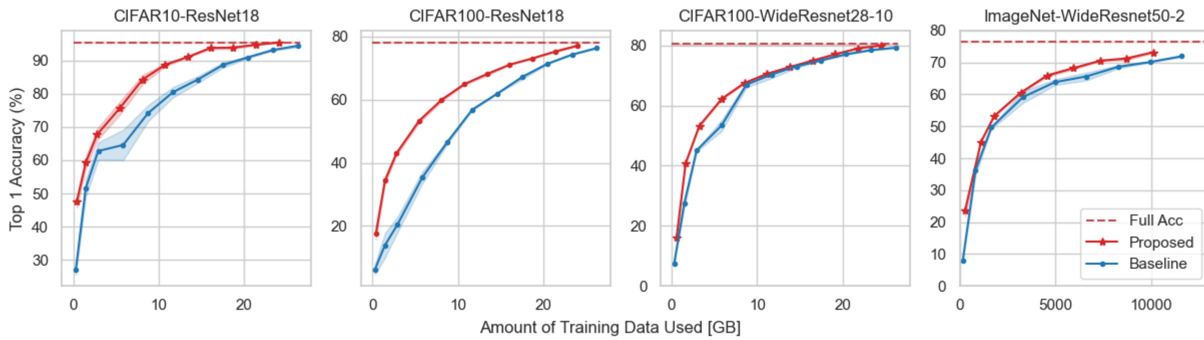


*Figure 2.* Top 1 Accuracy (%) improvement vs. increasing amount of data used for the training with the proposed framework applied to different models and datasets. From left to right, each point corresponds to training with 1%, 5%, 10%, 20%, 30%, ..., 90% subset of data, and corresponding data usage is converted to GB. CIFAR10/CIFAR100 dataset size is 135.82MB, and the Imagenet dataset size is 138GB. We used Glister (Killamsetty et al., 2021b) for our baseline subset selection.

*Table 3.* Ablation Study (Top-1 Accuracy (%))

| Method | Low fraction (10%) | | | | High fraction (70%) | | | |
|---|---|---|---|---|---|---|---|---|
| | ResNet18 CIFAR10 | ResNet18 ImageNet | WRN28-10 CIFAR100 | WRN50-2 ImageNet | ResNet18 CIFAR10 | ResNet18 ImageNet | WRN28-10 CIFAR100 | WRN50-2 ImageNet |
| Full acc | 95.4 | 67.7 | 80.4 | 76.5 | 95.4 | 67.7 | 80.4 | 76.5 |
| ❶ | 60.7±5.1 | 45.0±0.2 | 38.6±1.0 | 47.5±0.7 | 90.5±0.5 | 63.9±0.2 | 73.3±0.6 | 72.0±0.1 |
| ❶ + ❷ | 62.2±3.6 | 45.2±0.2 | 38.7±0.9 | 48.2±0.5 | 90.7±0.5 | 64.1±0.1 | 73.9±0.3 | 72.4±0.1 |
| ❶ + ❸ | 66.3±2.1 | 46.2±0.1 | 43.9±0.4 | 48.9±0.1 | 92.6±0.4 | 64.6±0.0 | 75.2±0.1 | 73.1±0.0 |
| ❶ + ❷ + ❸ | 68.5±1.1 | 46.4±0.2 | 45.1±0.3 | 49.4±0.2 | 93.6±0.3 | 64.7±0.1 | 76.4±0.5 | 73.3±0.0 |

❶ Model merging ❷ Model pruning ❸ Data Augmentation

ment is given in the appendix) – and this has a significant impact without data augmentation. To elaborate, when 10 jigsaw models obtain 1/5/10 batches without replacement, the overall data usage corresponds to 5% / 25% / 2% of the full dataset for CIFAR10 / CIFAR100 / ImageNet, respectively. Using such small amount of data is efficient and can be distributed-friendly, but can result in a data-scarce environment for each jigsaw learner. The result shows that the impact of the lack of data can be mitigated with data augmentation without additional resource cost (no additional real data is needed, so there is no extra I/O cost).

In our empirical study, we observed that the intermediate accuracy right after the pruning at each weak jigsaw learner is quite low at each worker. However, because the weak learners help improve diversity and robustness when multiple of them are merged the accuracy recovers significantly (to the near levels shown in the table), the variance is also lowered, and the final accuracy (after the main training) is also good as a result.

**Implications**. Our work has important implications for training models on large data sets. Viewing training from a systems perspective, two typical bottlenecks are storage (loading and processing data) and network communication (communication between workers). Our work shows that you can use extra computation (typically cheaper and more plentiful than storage or network bandwidth) to reduce or eliminate these bottlenecks; our framework uses data augmentation and model pruning (both computational tasks) to avoid processing the full subset of data, and uses ensembles and model merging to avoid network communication. Over the past five years, growth in total computational power (using GPUs) has been over 30×, while storage and network bandwidth has grown by less than one order of magnitude. Converting storage and network bottlenecks into computational tasks will prove useful in such a landscape.

### 4.5. Hyperparameter Sensitivity

As explained in Section 3, we have three important hyperparameters: the number of weak jigsaw models ($n$), the number of minibatches used for pretraining for each jigsaw model ($M$), and the number of steps each jigsaw model trains ($K$). We study sensitivity to these hyperparameters by measuring the Top 1 accuracy (%) of ResNet18 trained with a 10% subset of CIFAR100 using our framework. Figure 3 shows the sensitivity to different hyperparameters. In general, the cost of pretraining increases monotonically with each hyperparameter; a system would pick choices for these hyperparameters based on the overall I/O cost budget. Our results in this section show how such choices can be made and their influence on accuracy.

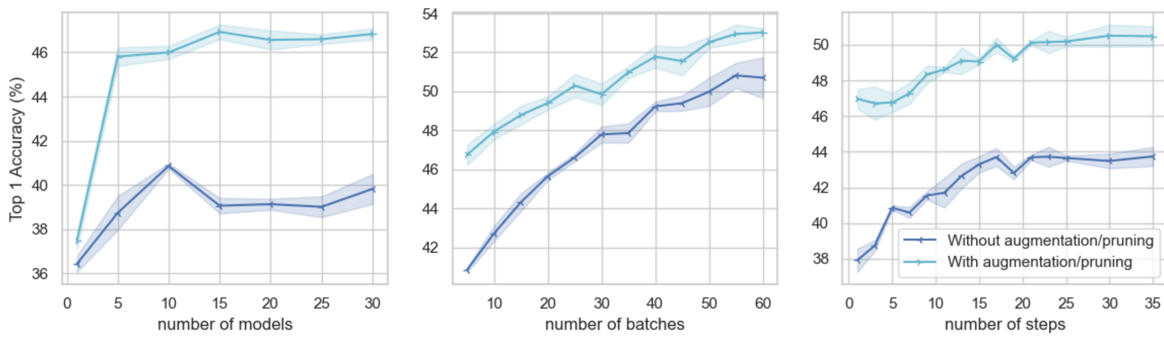


*Figure 3.* ResNet-18 trained with different hyperparameters with a 10% subset of CIFAR100. With increasing batches and steps, top-1 accuracy (%) increases so does the cost. For completeness, we also show trends without our techniques.

The leftmost figure shows the accuracy versus the number of jigsaw models. Interestingly, we observe a non-monotonic relationship between the numbers of models used to average the pre-trained model, and the final achieved accuracy after the main training. If we view the latter as a surrogate of the subset quality (and therefore, the quality of the pre-trained model's gradients), that tells us "the more the better" is not necessarily a true quote for our customized "model soup" ensembling for the pre-training purpose. Similar "non-monotonic" trend was previously observed in classical ensemble methods too (Zhou, 2012). Specifically, in our case, we conjecture the following factors to be accountable: our "model soup"-inspired averaging is a special ensembling for neural networks that aggregate weights, not predictions as commonly done. Meanwhile, convolutional neural networks are known to have a highly skewed/nearly sparse distribution of their parameter magnitudes, which are considered as necessary to encode the hidden low-dimensional structures of image features (Papyan et al., 2017) - that is further reinforced in our pre-training stage when each worker explicitly and independently enforces a sparse mask. Therefore, when we directly average too many neural networks (each trained from scratch, unlike (Wortsman et al., 2022b) which relies on a pre-trained model as a common "anchor") in their weight spaces, their vastly different sparse patterns in parameters will inevitably conflict with and compromise each other, leading to "over-smoothed" averaged weights that no longer preserves a meaningful sparse parameter/feature structure. Besides, prior works have also found that just increasing ensemble diversity, without maintaining the prediction accuracy of individual members, can quickly harm the ensemble performance (Nam et al., 2021). That explains why we cannot afford too high sparsity ratios nor more aggressive data augmentation in local training.

Next, we study the effect of the number of minibatches each weak jigsaw model trains on; the result is shown in the middle figure. In this experiment, we used 256 as the batch size and the CIFAR100 dataset. The full dataset contains 195 such batches. As expected, the more the minibatches, the greater the final accuracy – due to less likely overfitting as more data is used to train. Finally, the rightmost figure shows the effect of the number of steps ($K$) for each jigsaw model. We find that the accuracy plateaus after 10 steps, as each jigsaw model is trained with a limited dataset (of $M$ minibatches for a small $M$). Because there is a possibility of overfitting from using a large number of steps, we limit the maximum number of steps to 10 in all experiments.

**Implications**. Increasing the values of any hyperparameters mentioned above leads to an increase in latency, memory usage, and I/O cost. Unlike other hyperparameters in other training systems that require tuning via hyperparameter search, the hyperparameters in our framework are easily determined based on the overall system I/O cost budget. For our experiments, we set the number of models to be 10, constrain the number of minibatches to be less than 10, and the number of steps to be less than 10 as well for all datasets. Even with these conservative settings that yield low resource and I/O footprints, our accuracies could outperform the baselines.

## 5. Conclusion

We presented a lightweight, distribution-friendly pre-training framework for gradient-based subset training. Although training with a carefully selected subset showed promise to reduce training time and resources, prior work mainly focused on reducing the "main cost", while overlooking the "tax" at the essential pre-training stage. To reduce the pre-training tax and make the subset training more scalable, our distributed pre-training leveraged model-soup based ensembling to fully eliminate the communication between workers during the pre-training. The ensembling approach is also accompanied by data-driven pruning and random augmentation effectively to boost the diversity of the ensemble while reducing local overfitting, thereby providing robust gradients for subsequent subset selection algorithms.

## References


Agarwal, P., Har-Peled, S., and Varadarajan, K. Geometric approximation via coresets. combinatorial and computational geometry (je goodman, j. pach, and e. welzl, eds.). *Math. Sci. Research Inst. Pub., Cambridge*, 6:42, 2005.

Ainsworth, S. K., Hayase, J., and Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Breiman, L. Bagging predictors. *Machine learning*, 24(2): 123–140, 1996.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Chen, T., Cheng, Y., Gan, Z., Liu, J., and Wang, Z. Data-efficient gan training beyond (just) augmentations: A lottery ticket perspective. *Advances in Neural Information Processing Systems*, 34:20941–20955, 2021.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pp. 3259–3269. PMLR, 2020.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL `https://arxiv.org/abs/2101.00027`.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Guo, C., Zhao, B., and Bai, Y. Deepcore: A comprehensive library for coreset selection in deep learning. *arXiv preprint arXiv:2204.08499*, 2022.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 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.

Iyer, R., Khargoankar, N., Bilmes, J., and Asanani, H. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pp. 722–754. PMLR, 2021.

Killamsetty, K., Durga, S., Ramakrishnan, G., De, A., and Iyer, R. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pp. 5464–5474. PMLR, 2021a.

Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., and Iyer, R. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8110–8118, 2021b.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pp. 491–507. Springer, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, M., Gururangan, S., Dettmers, T., Lewis, M., Althoff, T., Smith, N. A., and Zettlemoyer, L. Branch-train-merge: Embarrassingly parallel training of expert language models. *ArXiv*, abs/2208.03306, 2022.

Maalouf, A., Jubran, I., and Feldman, D. Fast and accurate least-mean-squares solvers. *Advances in Neural Information Processing Systems*, 32, 2019.

Mirzasoleiman, B., Bilmes, J., and Leskovec, J. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pp. 6950–6960. PMLR, 2020.

Nam, G., Yoon, J., Lee, Y., and Lee, J. Diversity matters when learning from ensembles. *Advances in neural information processing systems*, 34:8367–8377, 2021.

Papyan, V., Romano, Y., and Elad, M. Convolutional neural networks analyzed via convolutional sparse coding. *The Journal of Machine Learning Research*, 18(1):2887–2938, 2017.

Paul, M., Ganguli, S., and Dziugaite, G. K. Deep learning on a data diet: Finding important examples early in training. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 20596–20607. Curran Associates, Inc., 2021.

Ramé, A., Ahuja, K., Zhang, J., Cord, M., Bottou, L., and Lopez-Paz, D. Recycling diverse models for out-of-distribution generalization. *arXiv preprint arXiv:2212.10445*, 2022.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Sinha, S., Zhang, H., Goyal, A., Bengio, Y., Larochelle, H., and Odena, A. Small-gan: Speeding up gan training using core-sets. In *International Conference on Machine Learning*, pp. 9005–9015. PMLR, 2020.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

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

Tripathi, S., Hemachandra, N., and Trivedi, P. Interpretable feature subset selection: A shapley value based approach, 2020. URL `https://arxiv.org/abs/2001.03956`.

Varma, M., Chen, X., Zhang, Z., Chen, T., Venugopalan, S., and Wang, Z. Sparse winning tickets are data-efficient image recognizers. In *Advances in Neural Information Processing Systems*.

Wang, X., Kondratyuk, D., Christiansen, E., Kitani, K. M., Alon, Y., and Eban, E. Wisdom of committees: An overlooked approach to faster and more accurate models. *arXiv preprint arXiv:2012.01988*, 2020.

Wei, K., Iyer, R., and Bilmes, J. Fast multi-stage submodular maximization. In *International conference on machine learning*, pp. 1494–1502. PMLR, 2014a.

Wei, K., Liu, Y., Kirchhoff, K., and Bilmes, J. Unsupervised submodular subset selection for speech data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4107–4111. IEEE, 2014b.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pp. 1954–1963. PMLR, 2015.

Wortsman, M., Gururangan, S., Li, S., Farhadi, A., Schmidt, L., Rabbat, M., and Morcos, A. S. lo-fi: distributed fine-tuning without communication. *arXiv preprint arXiv:2210.11948*, 2022a.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022b.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL `http://arxiv.org/abs/1605.07146`.

Zhou, Z.-H. *Ensemble methods: foundations and algorithms*. CRC press, 2012.