Dataset Efficient Training with Model Ensembling

Yeonju Ro[1,2], Cong Xu[2], Agnieszka Ciborowska[2], Suparna Bhattacharya[2], Frankie Li[2], Martin Foltin[2]
[1]The University of Texas at Austin, [2]Hewlett Packard Labs
{yeonju.ro, cong.xu, martin.foltin}@hpe.com

Abstract

*We propose a dataset-efficient deep learning training method by ensembling multiple models trained on different subsets. The ensembling method leverages the difficulty level of data samples to select subsets that are representative and diverse. The approach involves building a common base model with a random subset of data and then allotting different subsets to the models in an ensemble. The models are trained with their own subsets and then merged into a single model. We design an multi-phase training strategy that aggregates models in the ensemble more frequently and prevents divergence. The experiments on ResNet18 and ImageNet show that ensembling outperforms the no-ensemble case and achieves **64.8%** accuracy with **only 30% of dataset**, saving 20 hours of training time in a single V100 GPU training experiment with a mild accuracy drop.*

1. Introduction

Deep learning has revolutionized computer vision and other domains with ever-growing DNN architectures and training datasets, which promise remarkable performance at the price of prohibitively high training cost and energy consumption [10]. Multiple approaches focusing on different aspects of training have been proposed to increase the efficiency of the training procedure and democratize access to large DNNs. We have seen a growing number of tools supporting model and data distribution to multiple devices [1, 17], while others propose more efficient model optimization algorithms [11,25]. Recently, many researchers have focused on addressing training costs through data-centric approaches that explore using subsets of training data instead of the entire dataset [6, 13, 15, 21]. While these methods have shown promising results, they often need a carefully designed training procedure [8]. Additionally, some of these methods work well on certain datasets but fail on others [20], making them difficult to generalize across a wide range of applications.

To address these limitations, recent work [14] explored including samples of all difficulty levels in the training subset, where difficulty is defined as prediction uncertainty for the sample. The results indicated that such training subsets achieve higher accuracy, which is not surprising, as selecting data items from different difficulty levels naturally widens the selection space and makes models more general compared to focusing only on the most difficult examples [15, 21], which are closer to the classification boundaries. However, the particular combination of difficulty levels that works best can differ for different model and dataset characteristics. For instance, recent work [20] observed that keeping easy examples is better when the training data is scarce, while keeping hard examples is better when the training data is abundant.

This work presents a dataset-efficient training method by ensembling models trained on subsets of different difficulty levels. We first sort data samples based on the GraNd [15] algorithm, which scores each item based on its expected loss gradient norm. High-scoring samples are difficult, as their expected loss is high, while low-scoring samples are easy. Each model is trained with samples of different difficulty and then aggregated to generate one final model. By using the ensemble method, *we eliminate the need for experimentation to select the most appropriate sample difficulty for different architectures and datasets.* We also extend the method to an iterative multi-phase ensemble training method that aggregates models in the ensemble more frequently. Our results show that the proposed method outperforms the no-ensemble case. From the experiment with ResNet18 and the Imagenet Dataset, we achieved *64.8%* accuracy with *only 30% of the dataset* while the roofline accuracy is 67.65%.

2. Background

2.1. Subset training

Several previous works have proposed novel subset selection algorithms based on various factors to select the most effective samples without negative impact on the accuracy of a model trained with the subset. For example, some algorithms consider dataset geography [19,22], while others take into account error or loss [15,21]. Another approach is to use gradient matching [8,13]. In these approaches, a subset is selected to reduce the difference between the gradient from the subset and the gradient from the entire dataset.

These existing methods have primarily focused on providing theoretical guarantees for the quality of models. Thus, their experiments are usually limited to small datasets. However, using a single algorithm can generate a biased subset because of 1) a lack of representativeness and randomness, and 2) a model that can overfit to the subset. In comparison to these prior works, our approach takes a different angle: we focus on providing a practical method that can be applied to large datasets such as Imagenet.

The data pruning approach in [20] trains a probe student perceptron on the training data for a few epochs, computes the margin of each training example, constructs a subset of a certain size by retaining the hardest examples, and then trains a new perceptron to completion on the subset. Additionally, the authors used self-supervised learning (SSL) to develop a new, inexpensive unsupervised data pruning metric that does not require labels, unlike prior metrics.

2.2. Ensemble methods

Ensembling methods combine the predictions from multiple models to improve overall performance. Bagging [2] is a type of ensembling that trains a large number of strong learners (models that are relatively unconstrained) in parallel on different subsets of the training data. By averaging or voting the predictions of these learners, bagging can reduce variance and prevent overfitting of complex models.

Recently, researchers have studied the effectiveness of averaging *the parameters* instead of the predictions [4, 12, 23, 24]. [4] observed a connection between the optima of loss functions, enabling better ensembling methods. Based on this observation, *averaging the model parameters* is preferred over averaging predictions for some use cases. It was also used in federated learning [9] to synchronize models trained in different sites.

3. Proposed Methods

3.1. Ensemble of Models

Studies have shown the effectiveness of subset selection based on sample *difficulty* [3, 15, 21]. Intuitively, when a model is trained on difficult samples, it learns to generalize better and achieve higher performance. However, for GraNd, only data with a single level of difficulty is exposed to the entire training, restricting the representativeness of the subset and making it prone to overfitting. To address this issue, prior work [14] leveraged active learning to update the subset during training. However, this process requires more training epochs compared to baseline subset training since it gradually adds data samples to the subset.

To overcome the limitations of subset training with a lack of data diversity, we propose an ensemble of models, where each model is trained with samples of different difficulty. Figure 1 shows the high-level view of our approach, which consists of three main stages: (1) building a base model, (2) training an ensemble, and (3) model merging. First, we start with explaining the single-phase method (i.e., $n\_phase = 1$) where each stage is executed once, and, next, we extend the method to iterative multi-phase ensemble training in Section 3.2.

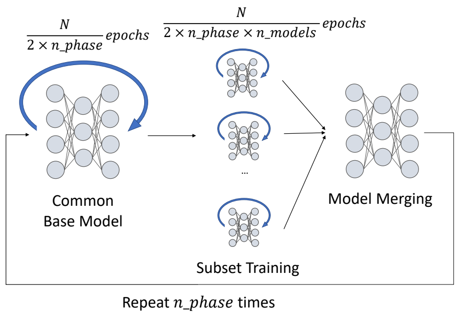
$\frac{N}{2 \times n\_phase}$ epochs
$\frac{N}{2 \times n\_phase \times n\_models}$ epochs
Common Base Model
...
Model Merging
Subset Training
Repeat n_phase times

Figure 1. Proposed Subset Training Method

In the first stage we train a common base model using a fixed subset of data, i.e., for $\frac{N}{2 * n\_phase}$ epochs, we do not re-select the subset. Here, N indicates the total budget for training (e.g., number of training epochs). To build the fixed subset, we select $k\%$ of samples using either a random sampling or a data selection algorithm. We will discuss the performance of both methods in the evaluation section. After the base model training is completed, we copy the base model n_models times to create an ensemble of size n_models, and assign each model a different subset to train on. To create subsets for ensemble models, we score all data samples in the full datasets using GraNd algorithm [15] and divide it into n_models difficulty classes. Next, from each difficulty class, we select $k\%$ data to form a final training subset. Each model in the ensemble is trained for $\frac{1}{n\_models}$ of the remaining training budget, and after the subset training is done, all models are merged into one single model using simple yet effective parameter averaging method [4, 16, 24].

3.2. Iterative Multi-phase Ensemble Training

Training ensemble models on separate subsets for an extended number of epochs can cause the models to diverge due to differing data distributions in each subset. This makes it challenging to maximize the benefits of weight averaging. To address this issue, we propose iterative multi-phase ensemble training. In addition to the single-phase ensemble training, this approach repeats the three stages described above for a specified number of phases (n_phases), with the training budget (e.g., the number of epochs) divided evenly between phases. By aggregating models more frequently, we can prevent the ensemble models from diverging too far from each other. In addition, we found that introducing the random subset in an interleaved fashion further improves training convergence.

For instance, if the training budget is 200 epochs, then single-phase ensemble training uses 100 epochs for base model training and the remaining 100 epochs for ensemble training where each ensemble gets a training budget of $\frac{100}{n\_models}$ epochs. Suppose we consider the same budget for the iterative multi-phase ensemble training with 2 phases. Then, each phase is assigned a budget of 100 epochs, with 50 epochs for training on a fixed subset and 50 epochs used for ensemble training. This effectively shortens the time models train on their unique subsets and should prevent subsets overfitting and, thus, diverging from the general data distribution.

4. Experiment Results

To evaluate the effectiveness of the proposed methods, we use ResNet18 [7] and ImageNet dataset [18] that includes 1.28 million training samples and 100,000 test samples. Before starting the proposed subset training method, we perform an initial model warm-up by training the model with a full dataset for 5 epochs with monotonically increasing learning rate from 0.1 as suggested by [5]. Each experiment has a total training budget of 200 epochs, and the models are trained with batch size of 256 and SGD optimizer. We use a cosine decay scheduler for learning rate scheduling. The initial learning rate is 0.1, the momentum is 0.9, and the weight decay is $5 * 1e - 4$.

4.1. Common Base Model

This experiment compares two methods for fixed subset selection and their effects on the performance of the base model. The *Top*-subset is created using the GraNd algorithm, which selects the most informative samples from the dataset based on gradient loss. The *Random*-subset is created by random sample selection.

Table 1. Accuracy for ensemble size=4 when the base model is trained on Top-subset vs Random-subset.

Data Fraction (k%)	1%	5%	10%	30%
Top	30.58	46.14	52.61	61.16
Random	**38.58**	**52.46**	**56.49**	**61.71**

Results in Table 1 shows that *Random*-subset-trained base model outperforms the *Top*-subset-trained base model. This is because *Top*-subset tends to include difficult samples that are more biased than *Random*-subset. This experiment result supports a claim that it is better to learn more diverse and general features during the early phase of training. Thus, for the remaining experimental results, we use *Random*-subset-trained model as our base model.

Figure 2 depicts the test loss measured at each epoch (the x-axis is converted to steps) for 10% and 30% subset training with a *Top*-subset base model and a *Random*-subset base model. For both 10% and 30% subsets, training with random subsets shows lower test loss.

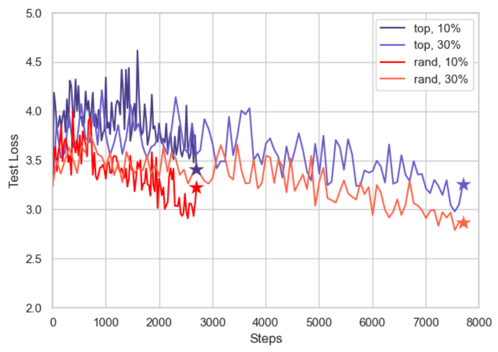

Figure 2. Test Loss for different base model

4.2. Effectiveness of Ensembling

Results in Table 2 present the performance of ensembling in a single-phase training. After we build a common base model, we perform *Ensemble of models* training, i.e., we continue the training of multiple models in parallel with different subsets. We compare the performance between ensembles of different sizes, where ensemble size is equal to the number of models n_model. Each model in the ensemble is trained for $\frac{100}{n\_models}$ epochs (note that the first 100 epochs are spent on training the common base model). To select a subset for each model, we sort all samples in the dataset by GraNd score, and divide it to n_model groups, and select $k\% \in \{1\%, 5\%, 10\%, 30\%\}$ data samples from each group. For a comparison, we added a *No ensemble* case, where the model continues training on the same subset that was used for the first 100 epochs.

Table 2. Accuracy depending on different ensemble sizes.

Data Fraction (k%)	1%	5%	10%	30%
No ensemble	18.10	43.53	49.92	57.98
Ensemble size = 2	35.56	48.16	53.56	60.52
Ensemble size = 4	**38.58**	52.46	56.49	**61.71**
Ensemble size = 10	38.31	**52.79**	**57.23**	60.90

Table 2 shows that *Ensemble of models* outperforms *No ensemble* training across all training subset sizes. We observe the highest increase in performance for the smallest subset size (i.e., $k = 1\%$). In contrast, when the subset size grows, the performance gap between *Ensemble of models* and *No ensemble* is reduced. This result indicates that *Ensemble of models* is particularly useful in the context of very small training subsets since each model in the ensemble gets the most representative fraction of samples from different difficulty classes and specializes in learning features distinctive to that class.

We also observe that accuracy for each data fraction reaches its plateau at a certain ensemble size. For instance, increasing the ensemble size from 2 to 4 improved the performance while increasing the size from 4 to 10 had a minor impact on the accuracy. The diminishing improvement can be explained by a fixed training budget, which is divided evenly by all the models in the ensemble. In other words, if as the ensemble size increases, each model in the ensemble is given less epochs to train on its subset. Thus, increasing the ensemble size without adjusting the training budget may bring limited improvements.

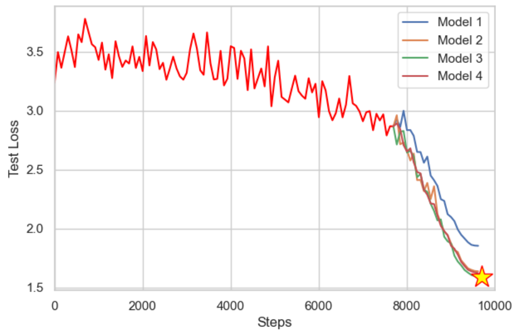

Figure 3. Test Loss for ensemble models size of four. Loss measured on the final aggregated model is depicted with a star.

To get more evidence on ensemble effectiveness, in Figure 3 we show the test loss measured at each epoch (the x-axis is converted to steps) for $k = 30\%$ subset training. We used an ensemble consisting of 4 models trained on top of a random-subset trained base model. For the remaining budget of 100 epochs, each model in the ensemble trains with a different subset for each $\frac{100}{4} = 25$ epochs, thus decreasing the loss accordingly. If we aggregate the models in the ensemble, the final loss (depicted with a star in the figure) is even lower than any of the models in the ensemble. This shows the effectiveness of the ensemble method.

4.3. Effectiveness of Iterative Multi-phase Ensemble Training

In this experiment, we evaluate the effectiveness of iterative multi-phase ensemble training proposed in Section 3.2. For iterative multi-phase ensemble training, entire training is divided into multiple phases. We measured the final accuracy for two configurations: when the number of phase is 2 and 5. For each phase, we do base model training and aggregation. When the number of phase is increased, we repeat the process of base model training and aggregation while number of epochs for total training is fixed to 200 epochs.

The experiment results show that if we increase the number of phases, accuracy rises only when the ensemble size is 2. In other words, increasing the number of phases does not always lead to better accuracy when the ensemble size is large. Intuitively, if we aggregate more frequently, the parameters of all the models in the ensemble should align with each other. However, if we aggregate too often, each model can train only for a limited number of epochs; hence, models cannot reach enough diversity. This limits the accuracy improvement in large ensemble and many phases.

Table 3. Accuracy of iterative multi-phase ensemble training.

Number of Phases	Ensemble Size	Data Fraction (k%) 1%	5%	10%	30%
2	2	37.78	52.09	57.47	63.78
	4	**39.68**	**53.33**	**58.52**	63.91
	10	37.98	52.29	57.94	63.28
5	2	38.53	52.85	58.24	**64.80**
	4	39.37	52.71	58.47	64.71
	10	35.58	50.86	57.10	64.24

Table 4. Accuracy and training time when model is trained with different data fractions.

	Data Fraction (k%) 1%	5%	10%	30%	Full
Accuracy	39.68	53.33	58.52	63.91	67.65
Time (hrs)	8.13	12.60	17.32	41.45	60.03

Finally, in Table 4 we examine the training efficiency of the proposed approach and compare it with the training time and accuracy of a model that is trained on the full dataset. Due to time constraints, the full dataset training was conducted for 100 epochs, while the ensemble training had a training budget of 200 epochs. For the comparison, we selected ensemble of size 4 with 2 phases. For the training time measurement, we used 1 V100 GPU. Overall, we observe that increasing the subset size improves accuracy at the cost of longer training time. Nonetheless, by selectively using 30% of the original data, the proposed method can significantly decrease the accuracy gap to the full dataset training while saving about 20 hrs in training time with minimal loss of accuracy.

5. Conclusion

In this paper, we proposed a data-efficient training method based on ensemble training. The proposed method independently trains multiple models with different subsets and merge all the models in the ensemble to build one final model. We also extended the method to an iterative multi-phase training, which allows early aggregation of models in the ensemble. Experiments show the proposed method outperforms the no-ensemble baseline, particularly when training with a small fraction of the full dataset.

References

[1] Zhengda Bian, Hongxin Liu, Boxiang Wang, Haichen Huang, Yongbin Li, Chuanrui Wang, Fan Cui, and Yang You. Colossal-ai: A unified deep learning system for large-scale parallel training. *CoRR*, abs/2110.14883, 2021. 1

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

[3] Kashyap Chitta, José M Álvarez, Elmar Haussmann, and Clément Farabet. Training data subset search with ensemble active learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14741–14752, 2021. 2

[4] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018. 2

[5] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 3

[6] Chengcheng Guo, Bo Zhao, and Yanbing Bai. Deepcore: A comprehensive library for coreset selection in deep learning. *arXiv preprint arXiv:2204.08499*, 2022. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[8] Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021. 1

[9] Jakub Konecný, H. B. McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *ArXiv*, abs/1511.03575, 2015. 2

[10] Andrey Kurenkov. Gpt-3 is no longer the only game in town, Nov 2021. 1

[11] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning, 2020. 1

[12] Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *ArXiv*, abs/2208.03306, 2022. 2

[13] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020. 1

[14] Dongmin Park, Dimitris Papailiopoulos, and Kangwook Lee. Active learning is a strong baseline for data subset selection. In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022. 1, 2

[15] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20596–20607. Curran Associates, Inc., 2021. 1, 2

[16] Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. Recycling diverse models for out-of-distribution generalization. *arXiv preprint arXiv:2212.10445*, 2022. 2

[17] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. 1

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 3

[19] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017. 1

[20] Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*, 2022. 1, 2

[21] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018. 1, 2

[22] Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128, 2009. 1

[23] Mitchell Wortsman, Suchin Gururangan, Shen Li, Ali Farhadi, Ludwig Schmidt, Michael Rabbat, and Ari S Morcos. lo-fi: distributed fine-tuning without communication. *arXiv preprint arXiv:2210.11948*, 2022. 2

[24] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022. 2

[25] Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. 1