Explore and Establish Synergistic Effects Between Weight Pruning and Coreset Selection in Neural Network Training

Weilin Wan[1], **Fan Yi**[1], **Weizhong Zhang**[2*], **Quan Zhou**[1], **Cheng Jin**[1,3]
[1]College of Computer Science and Artificial Intelligence, Fudan University
[2]School of Data Science, Fudan University
[3]Shanghai Key Laboratory of Intelligent Information Processing
{wlwan23, fyi21}@m.fudan.edu.cn, weizhongzhang@fudan.edu.cn,
qzhou21@m.fudan.edu.cn, jc@fudan.edu.cn

Abstract

Modern deep neural networks rely heavily on massive model weights and training samples, incurring substantial computational costs. Weight pruning and coreset selection are two emerging paradigms proposed to improve computational efficiency. In this paper, we first explore the interplay between redundant weights and training samples through a transparent analysis: redundant samples, particularly noisy ones, cause model weights to become unnecessarily overtuned to fit them, complicating the identification of irrelevant weights during pruning; conversely, irrelevant weights tend to overfit noisy data, undermining coreset selection effectiveness. To further investigate and harness this interplay in deep learning, we develop a **S**imultaneous **W**eight **a**nd **S**ample **T**ailoring mechanism (SWaST) that alternately performs weight pruning and coreset selection to establish a synergistic effect in training. During this investigation, we observe that when simultaneously removing a large number of weights and samples, a phenomenon we term critical double-loss can occur, where important weights and their supportive samples are mistakenly eliminated at the same time, leading to model instability and nearly irreversible degradation that cannot be recovered in subsequent training. Unlike classic machine learning models, this issue can arise in deep learning due to the lack of theoretical guarantees on the correctness of weight pruning and coreset selection, which explains why these paradigms are often developed independently. We mitigate this by integrating a state preservation mechanism into SWaST, enabling stable joint optimization. Extensive experiments reveal a strong synergy between pruning and coreset selection across varying prune rates and coreset sizes, delivering accuracy boosts of up to 17.83% alongside 10% to 90% FLOPs reductions.

Introduction

Training modern deep neural networks requires huge computational power and storage due to the large amount of model weights and datasets. Weight pruning and coreset selection (Hoefler et al. 2021; Killamsetty et al. 2021a) are two emerging paradigms to enhance training efficiency or model quality. Their key ideas are to develop proper rules to remove redundant model weights and to select the most informative samples during training, respectively. Notably, in scenarios with noisy data, coreset selection has been shown to improve model accuracy by filtering out less valuable samples. It has been repeatedly reported in the literature (Hoefler et al. 2021; Mirzasoleiman, Bilmes, and Leskovec 2020; Killamsetty et al. 2021a) that the proposed algorithms under these two paradigms can effectively reduce training costs with negligible performance degradation.

However, we notice that weight pruning and coreset selection are always investigated independently and their interaction is overlooked. In contrast, for the two corresponding techniques i.e., feature screening [1] and sample screening, in classical machine learning such as support vector machines (Shibagaki et al. 2016; Zhang et al. 2017), their interaction has been extensively studied and exploited to develop efficient training acceleration algorithms. Theoretical analysis shows that there exists a significant synergistic effect between feature and sample screening, that is, discarding the identified redundant features (resp. samples) leads to a more accurate estimation of the primal (resp. dual) optimum, which in turn enhances the sample (resp. feature) screening rules in detecting irrelevant samples (resp. features). These results are derived mainly depending on the theoretical tools, e.g., Karush–Kuhn–Tucker (KKT) conditions (Kuhn and Tucker 2013), in convex optimization. Effective joint screening methods are proposed to achieve a synergistic effect, which enables model training with 10 million of features and samples possible on the resource-restricted devices such as laptop. *Although the above optimal tools do not exist in deep learning due to the highly complicated nonconvex training problem, there seems to be no reason to assert that a similar mechanism of the synergy effect does not hold in weight pruning and coreset selection paradigms, which is the main motivation of this work.*

The first thing that can be expected is that the coreset selection training process appears to be inherently more susceptible to overfitting and thus would benefit more from the regularization effects of weight pruning. To provide an in-depth and transparent analysis of the interaction between weight pruning and coreset selection in deep learning, and considering the complexity of deep neural networks, we first perform an in-depth investigation on polynomial interpolation tasks. We employ standard weight pruning and core-

*Corresponding Author.

[1]Features in classical machine learning models always have one-to-one correspondence with model weights.

set selection techniques for neural networks in this experiment (Fig. 1) to ensure that the results reflect deep learning principles. We get the following two key observations. One is that an excessive number of samples to be interpolated, especially noisy ones, can cause the coefficients of the polynomial to be overtuned in order to fit all the samples. This significantly complicates the identification of irrelevant weights during pruning, as most existing pruning methods develop their importance metric explicitly or implicitly based on the weight magnitude. The other observation is that, under the standard metric in coreset selection, the difficulty of selection grows exponentially with the number of redundant model weights. Our initial findings expose a critical interaction between model weights and training samples: redundant samples complicate weight pruning, while irrelevant weights obscure coreset selection. This mutual interference implies the limitations of traditional approaches that treat pruning and coreset selection as independent processes.

To further investigate and harness the above interplay in deep learning, we develop a **S**imultaneous **W**eight **a**nd **S**ample **T**ailoring mechanism (SWaST) that alternately performs weight pruning and coreset selection to establish a synergistic effect in training. Every $\mathcal{R}$ epochs, SWaST performs coreset selection to identify the most representative samples. During subsequent $\mathcal{R}$ epochs, it performs training and online pruning using this selected coreset. We derive two variants based on the aggressiveness of the pruning strategy: SWaST-trim (which prunes only the fully connected layers) and SWaST-cut (which prunes the entire network). SWaST-trim utilizes a conservative pruning strategy, retaining more parameters and ensuring better training stability. In contrast, SWaST-cut removes more parameters for greater efficiency, but this can lead to training instability, referred to as "*critical double-loss*". This instability occurs when pivotal weights and their supportive samples are mistakenly eliminated at the same time, causing an almost irreversible performance degradation. In contrast to classic machine learning models, this phenomenon can easily occur in deep learning due to the absence of theoretical guarantees in pruning and coreset selection. This underlying complexity elucidates why existing methods in pruning and coreset selection have traditionally been developed in isolation. To ensure training stability, we incorporate a state preservation mechanism that captures the model's output following coreset selection. This mechanism helps preserve critical knowledge during training and pruning while allowing adaptation to the refined dataset and architecture. Extensive experiments reveal a strong synergy between pruning and coreset selection across varying prune rates and coreset sizes, delivering accuracy boosts of up to 17.83% alongside 10% to 90% FLOPs reductions.

Our main contributions can be summarized as follows.

- We conduct a transparent analysis to explore and identify the fundamental interplay between weight pruning and coreset selection within deep learning.
- We propose an alternative pruning and coreset selection method SWaST, designed to further investigate and exploit the interplay to establish a synergistic effect.
- We design a state preservation mechanism for SWaST that stabilizes training and maintains model performance under aggressive pruning rates and small coreset sizes.
- Experiments verify the synergistic effect established by SWaST, with a significant accuracy gain of up to 17.83%.

Related Work

Coreset Selection

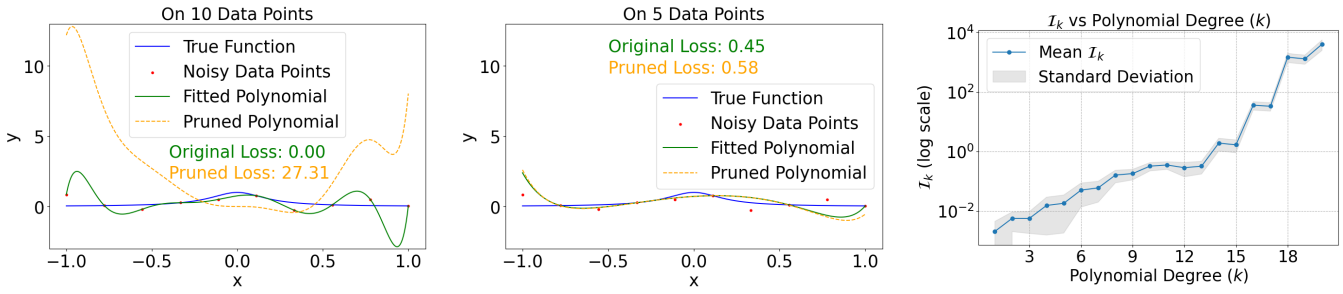
Coreset selection identifies a representative subset from the original (noisy) dataset with the objective of increasing training efficiency or enhancing model effectiveness. Recent methods are categorized into optimization-based and heuristic-based. Optimization-based approaches employ bilevel optimization (Borsos, Mutny, and Krause 2020; Zhou et al. 2022; Hao, Ji, and Liu 2024) or gradient matching objectives (Killamsetty et al. 2021a; Mirzasoleiman, Bilmes, and Leskovec 2020; Killamsetty et al. 2021b,c), providing theoretical guarantees, but incurring high costs. Heuristic-based methods assess sample importance via forgetting events (Toneva et al. 2018), gradient (Pruthi et al. 2020; Xia et al. 2024; Thakkar et al. 2023), and inter-sample distances (Xia et al. 2022). Recent progress includes DNN-based selectors (Ilyas et al. 2022; Engstrom, Feldmann, and Madry 2024) and efficient frameworks (Everaert and Potts 2023; Paul, Ganguli, and Dziugaite 2021; Xiao et al. 2024; Park et al. 2024) that balance quality and efficiency.

Weight Pruning

Weight pruning is a key technique for model compression and acceleration, classified into structured and unstructured methods. Structured pruning methods (Filters'Importance 2016; He et al. 2018; Jiang, Cao, and Yang 2022; Elkerdawy et al. 2022; Guan et al. 2022; Nonnenmacher et al. 2021; Alvarez and Salzmann 2016; Murray and Chiang 2015) remove entire filters or channels, enabling hardware acceleration but often reducing accuracy. In contrast, unstructured pruning (Tanaka et al. 2020; Lee, Ajanthan, and Torr 2018; Han, Mao, and Dally 2015; Frankle and Carbin 2018) eliminates individual weights based on metrics like magnitude or gradient sensitivity (Evci et al. 2020; Han et al. 2015; Wen et al. 2016; Blakeney, Yan, and Zong 2020; Renda, Frankle, and Carbin 2020; Hassibi and Stork 1992; LeCun, Denker, and Solla 1989). Recent theoretical work link pruning to optimization theory (Grosse and Martens 2016; Zhou et al. 2021; Srinivas, Subramanya, and Venkatesh Babu 2017; Louizos, Welling, and Kingma 2017; Qian and Klabjan 2021), bolstering its foundations.

Pruning and Coreset Selection in Classical ML

In classical machine learning, feature and sample screening mirror weight pruning and coreset selection. Optimization-based theoretical frameworks exist for joint feature and sample screening in this domain (Shibagaki et al. 2016; Zhang et al. 2017). Empirically, feature screening enhances sample importance estimation, and vice versa. While these theories don't translate directly to deep learning, this synergy motivates integrating weight pruning and coreset selection.



(a) On noisy samples (b) On clean samples (c) $\mathcal{I}(\mathcal{D}, \hat{\mathcal{D}})$ values

Figure 1: The impact of redundant weights and samples on pruning and coreset selection. The collapsed pruned model (yellow curve) in (a) compared to (b) implies that noisy/redundant data increases the difficulty of pruning. (c) shows selection difficulty $\mathcal{I}(\mathcal{D}, \hat{\mathcal{D}})$ rises with polynomial degree, indicating harder coreset selection.

Interplay of Redundant Samples and Weights

We explore the interplay between redundant weights and samples by investigating their impact on weight pruning and coreset selection. In order to provide mathematically transparent analysis, we apply standard pruning and coreset selection techniques to polynomial interpolation.

Impact of redundant samples on weight pruning. We study this impact on the polynomial interpolation task with a classic example Runge's function $f(x) = \frac{1}{1+25x^2}, x \in [-1, 1]$ (Runge 1901), which is smooth and has derivatives of all orders, as shown by the blue curve in Fig. 1a and 1b (see appendix for results on other functions). Our dataset comprises 10 points sampled from the function: 5 are clean data points (which form the coreset), and the other 5 are noisy points, generated by adding Gaussian noise with a magnitude of 0.1. Using the least squares method, we fitted a 10^{th}-degree polynomial to two datasets separately: the full dataset of 10 points and the 5-point coreset. Following the standard weight pruning techniques (Han, Mao, and Dally 2016), we pruned the three smallest coefficients (in magnitude) by setting them to zero while retaining the remaining coefficients. Our findings indicate that an increased number of data points, especially those containing noise, renders pruning more challenging. This is evidenced by the higher loss observed (yellow curve in Fig. 1a) after pruning, which suggests that coreset selection can enhance the effectiveness of pruning in scenarios involving redundant samples.

Impact of redundant weights on coreset selection. The goal of coreset selection is to identify a small subset of training data that approximates the objective function of the entire dataset throughout the parameter space. A typical formulation is constructed on function approximation, which involves finding a small subset $\hat{\mathcal{D}}$ from the training dataset $\mathcal{D}$ such that:

$$\frac{|\mathcal{L}(\boldsymbol{\theta}) - \hat{\mathcal{L}}(\boldsymbol{\theta})|}{\mathcal{L}(\boldsymbol{\theta})} \leq \epsilon, \text{ for any } \boldsymbol{\theta} \in \mathbb{R}^p, \tag{1}$$

where $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \ell(\boldsymbol{\theta}; x_i, y_i)$ is the objective function on the full dataset, and $\hat{\mathcal{L}}(\boldsymbol{\theta}) = \frac{1}{|\hat{\mathcal{D}}|} \sum_{i \in \hat{\mathcal{D}}} \ell(\boldsymbol{\theta}; x_i, y_i)$ is the objective function on the selected subset $\hat{\mathcal{D}}$. Here, ϵ is a small number that controls the approximation error. When ϵ is sufficiently small, the model performance learned from $\hat{\mathcal{L}}$ can be comparable to that learned from $\mathcal{L}$. In this context,

$$\mathcal{I}(\mathcal{D}, \hat{\mathcal{D}}) = \sup_{\boldsymbol{\theta}} \frac{|\mathcal{L}(\boldsymbol{\theta}) - \hat{\mathcal{L}}(\boldsymbol{\theta})|}{\mathcal{L}(\boldsymbol{\theta})}, \tag{2}$$

can be used to evaluate the difficulty of coreset selection. The reason is that given two fixed datasets $\mathcal{D}$ and $\hat{\mathcal{D}}$, if $\mathcal{I}(\mathcal{D}, \hat{\mathcal{D}})$ for network $f_1(\cdot, \boldsymbol{\theta})$ is larger than network $f_2(\cdot, \boldsymbol{\theta})$, it means that $f_1(\cdot, \boldsymbol{\theta})$ needs to choose a larger coreset $\hat{\mathcal{D}}$ to achieve the same accuracy with $f_1(\cdot, \boldsymbol{\theta})$, i.e., the coreset selection in $f_1(\cdot, \boldsymbol{\theta})$ is more difficult than $f_2(\cdot, \boldsymbol{\theta})$.

In our second polynomial experiment, we employed Runge's function to create samples and applied Gaussian noise with a strength of 0.1. We optimized the problem in Eq. (2) for polynomial degrees $k = 1, \ldots, 20$ and present the result in Fig. 1c. It shows that with the y-axis in logarithmic scale, $\mathcal{I}(\mathcal{D}, \hat{\mathcal{D}})$ grows exponentially with increasing polynomial degree. The reason is that $\mathcal{L}(\boldsymbol{\theta})$ vanishes to zero rapidly due to the overfitting of the noisy data. This suggests that coreset selection difficulty increases with model parameter dimension rapidly. Therefore, it can be expected that pruning can mitigate this issue by reducing model size.

The Proposed Training Mechanism

In this section, we propose SWaST to further investigate and exploit the interplay above to establish a synergistic effect between weights pruning and coreset selection.

A Synergistic Optimization Mechanism

SWaST consists of a warm-up phase and an alternating optimization phase (shown in Fig. 2). In the warm-up phase, the network is trained on the full dataset for $\mathcal{K}$ epochs to establish a good initialization. The subsequent phase alternatively performs weight pruning and coreset selection, where coreset selection is performed every $\mathcal{R}$ epochs to identify representative samples, followed by network training on the selected samples with pruning operations. This alternating

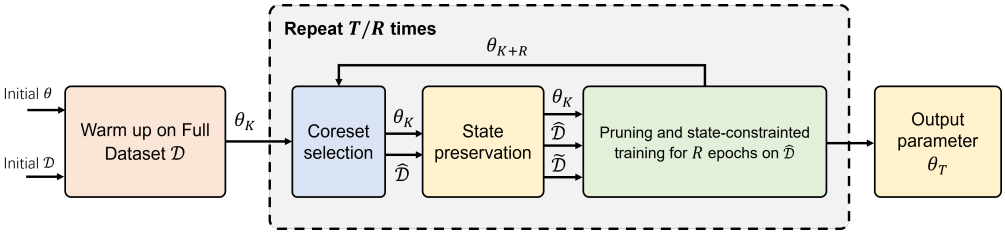


Figure 2: Overview of SWaST, alternating between pruning and coreset selection every $\mathcal{R}$ epochs. $\mathcal{D}$ denotes the full dataset, $\hat{\mathcal{D}}$ denotes the selected subset, $\tilde{\mathcal{D}}$ denotes the stored logits, and T denotes the total epochs.

process effectively removes redundant samples and parameters while maintaining model performance. The detailed procedure is outlined in Algorithm 1. We propose two variants that use different pruning strategies:

- **SWaST-trim** prunes only the FC layer, retaining most parameters to ensure training stability. The improved efficiency is mainly attributed to coreset selection.
- **SWaST-cut** prunes the entire network to achieve more aggressive efficiency gains, but may lead to training instability (which we refer to as the **"double-loss problem"**). We develop the state preservation mechanism to mitigate this issue in the following section.

Algorithm 1: SWaST

Require: Dataset $\mathcal{D}$, warm-up epochs $\mathcal{K}$, epoch interval $\mathcal{R}$ for coreset selection, coreset to full set ratio α, pruning algorithm $\mathcal{P}$, coreset selection algorithm $\mathcal{S}$, state preservation weight λ, use state preservation flag use_SP.
Ensure: Trained network parameters $\boldsymbol{\theta}$

Initialize network parameters $\boldsymbol{\theta}$
Set $\mathcal{X} = \mathcal{D}$
Initialize stored logits $\tilde{\mathcal{D}} = \{\}$
for $t = 1, 2, \ldots, T$ **do**
  if $t > \mathcal{K}$ and $t \mod \mathcal{R} = 0$ **then**
    $\mathcal{X} \leftarrow \mathcal{S}(\mathcal{D}, \boldsymbol{\theta}, \alpha)$ ▷ Execute coreset selection
    if use_SP **then**
      $\tilde{\mathcal{D}} \leftarrow \{(\mathbf{x}_i, \mathbf{z}_i) : \mathbf{z}_i = f_{\boldsymbol{\theta}_{pre}}(\mathbf{x}_i), \mathbf{x}_i \in \mathcal{X}\}$
    end if
  end if
  for each training iteration **do**
    Sample mini-batch $\mathcal{B}$ from coreset $\mathcal{X}$
    Compute cross-entropy loss $\mathcal{L}_{CE}$
    if $\mathbf{Z}$ is not empty **then**
      Compute $\mathcal{L}_{SP}$ using $\tilde{\mathcal{D}}$
      $\mathcal{L}_{\text{total}} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{SP}$ ▷ SWaST-cut
    else
      $\mathcal{L}_{\text{total}} = \mathcal{L}_{CE}$ ▷ SWaST-trim
    end if
    Compute gradients $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{total}}$
    $\boldsymbol{\theta} \leftarrow \mathcal{P}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{total}})$ ▷ Execute pruning step
    Update parameters $\boldsymbol{\theta}$ using gradient descent
  end for
end for
return $\boldsymbol{\theta}$

Remark 1 *SWaST maintains generality by supporting any pruning algorithm and coreset selection method that can be integrated into the training process.*

- *For pruning, the only requirement is that the method should be capable of operating in an online manner, updating the network structure during training rather than requiring separate pre-training or post-processing steps.*
- *For coreset selection, any method that evaluates and selects samples based on the current model is compatible.*

This flexibility allows practitioners to select the most suitable pruning and coreset selection approaches for their specific needs while still benefiting from SWaST's synergistic optimization between pruning and coreset selection.

Double-loss Problem: Analysis and Mitigation

We begin by establishing two baseline observations:

- For pruning only scenarios, when a parameter is incorrectly pruned, it can be recovered through learning from appropriate training samples in the next iteration.
- Similarly, for coreset only scenarios, when a training sample is mistakenly excluded, the knowledge preserved in the network parameters can help reidentify and select this sample in the next round.

However, when performing weight pruning and sample tailoring simultaneously, we may encounter situations in which both a parameter and its corresponding supportive training samples are eliminated together. In such cases, standard training procedures struggle to recover from this information loss, as neither component remains available to assist in the recovery process. We term this the "double-loss" problem. Fig. 3 illustrates this phenomenon through a representative pair of interrelated data and parameters.

Remark 2 *Fig. 3 offers an intuitive illustration of the double-loss phenomenon, rather than experimental or theoretical validation. It depicts how losing interconnected parameters and samples can impede training recovery. The actual parameter-sample interactions in neural networks occur in higher dimensions with greater complexity.*

Remark 3 *It is important to note that the critical double-loss phenomenon is unique to deep learning. Classical machine learning models leverage theoretical tools like KKT conditions (Shibagaki et al. 2016) for safe feature/sample removal. In contrast, deep models' highly non-convex loss function prevent such guarantees, necessitating dynamic stabilization mechanisms, which is developed below.*

To address this challenge while maintaining optimization stability, we propose **a state preservation mechanism** that operates through two alternating phases:

Stage 1: State recording (executed every $\mathcal{R}$ epochs). Following each coreset update, we capture the model's states via full forward propagation, i.e.,

$$\tilde{\mathcal{D}} = \{(\mathbf{x}_i, \mathbf{z}_i) : \mathbf{z}_i = f_{\boldsymbol{\theta}_{pre}}(\mathbf{x}_i), \mathbf{x}_i \in \mathcal{X}\}.$$

These preserved states encode critical information about the current model and data subset, including both explicit prediction patterns and feature correlations.

Stage 2: State-constrained training. During subsequent parameter updates, we enforce state consistency through the composite loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda \sum_{(\mathbf{x}_i, \mathbf{z}_i) \in \tilde{\mathcal{D}}} \text{KL}(\sigma(\mathbf{z}_i) \| \sigma(f_{\boldsymbol{\theta}}(\mathbf{x}_i))), \quad (3)$$

where λ (default 0.1) balances the primary learning objective with state consistency, σ denotes the softmax function, and KL represents the Kullback-Leibler divergence. Note that the KL terms in Eq.(3) imply that we utilize all samples in $\mathcal{X}$, as even misclassified examples provide valuable information about the model's optimization trajectory.

The mechanism above directly addresses the "double-loss" problem with the following capabilities:

- KL terms can correct mistakes from previous pruning steps. If a critical parameter is erroneously pruned, the KL term can detect this error in subsequent steps—since the KL divergence relative to the preserved state $\mathbf{z}_i$ will increase—and re-select that parameter.
- The KL terms help mitigate training instability caused by pruning. Mistakenly removing important parameters can lead to large fluctuations in $\sigma(\mathbf{z}_i)$ and destabilize training. Such fluctuations are detected by the KL term, which measures distributional divergence across all classes, whereas $\mathcal{L}_{\text{CE}}$ only penalizes the ground-truth class.

Experiments

To evaluate the effectiveness and robustness of SWaST, we conducted a series of experiments. All results reported in the tables are averaged over 5 runs. We first briefly review the experimental setup, followed by specific results. Detailed method descriptions and additional results with other pruning/coreset methods, can be found in the Appendix.

- **Datasets and network architectures:** We conduct experiments on CIFAR-10/100 (Krizhevsky, Hinton et al. 2009), TinyImageNet (Le and Yang 2015), and ImageNet-1K (Deng et al. 2009) datasets, using ResNet-18 and ResNet-101 to verify our method's effectiveness across different model scales.
- **Coreset selection and pruning methods:** Given the varying scales of datasets, we adopt selection methods with different computational complexities: GradMatch (Killamsetty et al. 2021a) for CIFAR, Moderate (Xia et al. 2022) for TinyImageNet, and EL2N (Paul, Ganguli, and Dziugaite 2021) for ImageNet-1K. We employ the RigL (Evci et al. 2020) method as our default pruning algorithm.

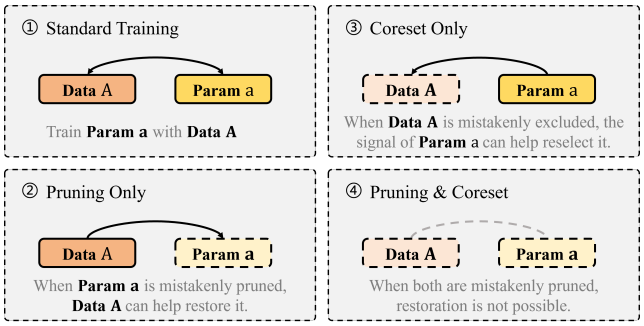


Figure 3: Illustration of the "critical double-loss" phenomenon in concurrent weight pruning and sample tailoring: (1) Standard Training shows the close link between Data A and Param a. (2) Pruning Only allows recovery of Param a using Data A when mistakenly pruned. (3) Coreset Only supports re-selection of Data A through Param a when excluded in error. (4) Pruning & Coreset causes irreversible degradation from simultaneous exclusion.

- **Pruning configuration:** For SWaST-trim, the pruning rate applies to FC layers; for SWaST-cut, it applies to the backbone with FC layers fixed at 90% pruning. Configuration details are provided in the Appendix. Detailed analysis and justification for these configuration choices are provided in the Appendix.
- **Evaluation on datasets with label noise:** To thoroughly analyze the synergistic effects between weight pruning and coreset selection, we introduce label noise into the datasets in certain experiments.

Main Results

Efficiency analysis. To validate the efficiency of SWaST, we present experimental results conducted on ResNet101, as illustrated in Fig. 4. In this visualization, the x-axis represents FLOPS, the y-axis shows accuracy, and the circle

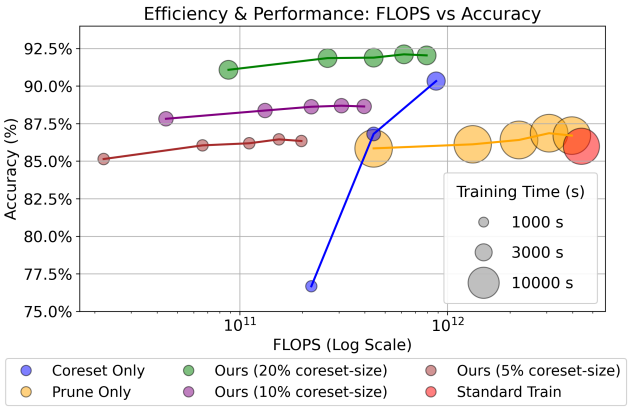


Figure 4: Efficiency vs. accuracy trade-offs (bubble area $\propto$ training time), where the upper-left region indicates optimal performance (high accuracy with low FLOPs).

Model	Dataset	Coreset Size	SWaST-trim with different Prune Rate					Coreset-only
			90%	70%	50%	30%	10%	
ResNet-18	CIFAR-10	10%	92.58$_{(+0.38)}$	92.53$_{(+0.33)}$	92.40$_{(+0.20)}$	92.32$_{(+0.12)}$	92.29$_{(+0.09)}$	92.20
		5%	90.16$_{(+0.49)}$	90.07$_{(+0.40)}$	89.92$_{(+0.25)}$	89.81$_{(+0.14)}$	89.77$_{(+0.10)}$	89.67
		1%	78.60$_{(+2.97)}$	78.33$_{(+2.80)}$	77.14$_{(+1.61)}$	76.26$_{(+0.73)}$	76.01$_{(+0.48)}$	75.53
	CIFAR-100	10%	71.94$_{(+0.97)}$	71.76$_{(+0.79)}$	71.44$_{(+0.47)}$	71.27$_{(+0.30)}$	71.20$_{(+0.23)}$	70.97
		5%	66.19$_{(+1.40)}$	65.97$_{(+1.18)}$	65.56$_{(+0.77)}$	65.22$_{(+0.43)}$	65.09$_{(+0.30)}$	64.79
		1%	38.38$_{(+1.96')}$	38.12$_{(+1.70)}$	37.58$_{(+1.16)}$	37.01$_{(+0.59)}$	36.88$_{(+0.46)}$	36.42
ResNet-101	CIFAR-10	10%	92.60$_{(+1.47)}$	92.45$_{(+1.32)}$	92.12$_{(+0.99)}$	91.89$_{(+0.76)}$	91.54$_{(+0.41)}$	91.13
		5%	89.81$_{(+3.53)}$	89.39$_{(+3.11)}$	88.04$_{(+1.76)}$	87.46$_{(+1.18)}$	87.29$_{(+1.01)}$	86.28
		1%	66.23$_{(+16.51)}$	63.36$_{(+13.64)}$	60.46$_{(+10.74)}$	55.45$_{(+5.73)}$	53.02$_{(+3.30)}$	49.72
	CIFAR-100	10%	71.93$_{(+3.14)}$	71.68$_{(+2.89)}$	70.47$_{(+1.68)}$	70.01$_{(+1.22)}$	68.74$_{(+0.95)}$	68.79
		5%	61.72$_{(+7.84)}$	60.25$_{(+6.37)}$	58.19$_{(+4.31)}$	56.63$_{(+2.75)}$	55.82$_{(+1.94)}$	53.88
		1%	21.16$_{(+9.47)}$	18.53$_{(+6.84)}$	16.04$_{(+4.35)}$	14.46$_{(+2.77)}$	13.87$_{(+2.18)}$	11.69
Model	**Dataset**	**Coreset Size**	**SWaST-cut with different Prune Rate**					**Coreset-only**
			90%	70%	50%	30%	10%	
ResNet-18	CIFAR-10	10%	91.53$_{(-0.67)}$	92.32$_{(+0.12)}$	92.40$_{(+0.20)}$	92.92$_{(+0.72)}$	92.60$_{(+0.40)}$	92.20
		5%	89.32$_{(-0.35)}$	90.12$_{(+0.45)}$	90.35$_{(+0.68)}$	91.11$_{(+1.44)}$	90.18$_{(+0.51)}$	89.67
		1%	77.13$_{(+1.60)}$	78.52$_{(+2.99)}$	78.57$_{(+3.24)}$	78.70$_{(+3.17)}$	78.67$_{(+3.14)}$	75.53
	CIFAR-100	10%	67.98$_{(-2.99)}$	71.09$_{(+0.12)}$	71.91$_{(+0.94)}$	72.09$_{(+1.12)}$	71.99$_{(+1.02)}$	70.97
		5%	63.24$_{(-1.55)}$	65.15$_{(+0.36)}$	65.55$_{(+0.76)}$	66.81$_{(+2.02)}$	66.30$_{(+1.51)}$	64.79
		1%	35.80$_{(-0.62)}$	37.81$_{(+1.39)}$	40.29$_{(+3.87)}$	39.55$_{(+3.13)}$	38.53$_{(+2.11)}$	36.42
Params. After Pruning			1.2M	3.5M	5.8M	8.2M	10.5M	11.7M
ResNet-101	CIFAR-10	10%	91.85$_{(+0.72)}$	92.46$_{(+1.33)}$	92.61$_{(+1.48)}$	92.96$_{(+1.83)}$	92.62$_{(+1.49)}$	91.13
		5%	88.90$_{(+2.62)}$	89.55$_{(+3.27)}$	90.00$_{(+3.72)}$	90.13$_{(+3.85)}$	89.99$_{(+3.71)}$	86.28
		1%	64.57$_{(+14.85)}$	64.82$_{(+15.10)}$	66.92$_{(+17.20)}$	67.55$_{(+17.83)}$	66.52$_{(+16.80)}$	49.72
	CIFAR-100	10%	69.41$_{(+0.62)}$	70.73$_{(+1.94)}$	71.75$_{(+2.96)}$	71.95$_{(+3.16)}$	72.05$_{(+3.26)}$	68.79
		5%	59.27$_{(+5.39)}$	59.65$_{(+5.77)}$	62.87$_{(+8.99)}$	62.97$_{(+9.09)}$	62.14$_{(+8.26)}$	53.88
		1%	19.17$_{(+7.48)}$	20.11$_{(+8.42)}$	21.24$_{(+9.55)}$	20.99$_{(+9.30)}$	21.19$_{(+9.50)}$	11.69
	TinyImageNet	10%	51.29$_{(+2.63)}$	52.78$_{(+4.12)}$	53.49$_{(+4.83)}$	53.14$_{(+4.48)}$	52.98$_{(+4.32)}$	48.66
		5%	38.65$_{(+3.55)}$	42.83$_{(+7.73)}$	42.93$_{(+7.83)}$	41.94$_{(+6.84)}$	41.21$_{(+6.11)}$	35.10
	ImageNet-1k	10%	37.55$_{(+5.83)}$	38.28$_{(+6.56)}$	38.92$_{(+7.20)}$	39.19$_{(+7.47)}$	39.47$_{(+7.75)}$	31.72
		5%	31.63$_{(+1.35)}$	32.25$_{(+1.97)}$	32.71$_{(+2.43)}$	34.34$_{(+4.06)}$	33.12$_{(+2.84)}$	30.28
Params. After Pruning			4.5M	13.3M	22.3M	31.2M	40.1M	44.5M

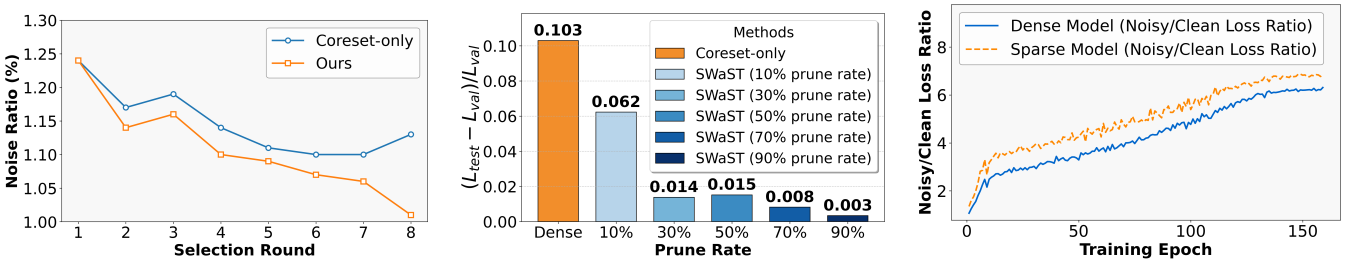
Table 1: Comparison for various pruning rates and subset fractions on different datasets. Values in parentheses show differences from the Coreset-only baseline. Red/blue numbers indicate best and runner-up results within each row. SWaST-trim (upper half) prunes only FC layers, maintaining parameters near originals: ∼11.2M–11.7M for ResNet-18 (original 11.7M) and ∼42.7M–44.5M for ResNet-101 (original 44.5M). SWaST-cut (lower half) applies global pruning with explicit parameter counts listed. **Row-wise**: Both SWaST variants consistently outperform the baseline, with gains up to **17.83%**. **Column-wise**: Performance improvements **increase as coreset sizes decrease** since smaller coresets present greater selection challenge and higher overfitting risk, while weight pruning mitigates these issues.

sizes indicate training time (detailed experimental configurations and settings are provided in the appendix). As observed from the results, our method achieves significantly higher accuracy compared to other approaches under equivalent FLOPS constraints, demonstrating superior computational efficiency.

Weight pruning improves coreset selection. Following the protocol established in (Killamsetty et al. 2021a), we benchmark our proposed methods (SWaST-trim and SWaST-cut) against the coreset-only baseline. Table 1 demonstrates that SWaST variants exhibit two key advantages when operating with identical subset sizes:

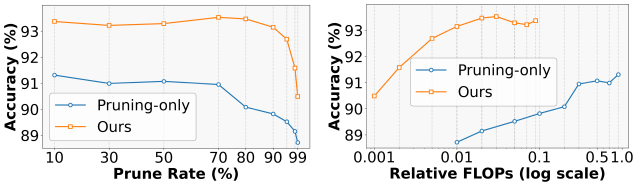
- **Row-wise analysis:** Both SWaST-trim and SWaST-cut consistently outperform the coreset-only baseline across all configurations, achieving accuracy improvements of up to 17.83%.
- **Column-wise analysis:** The performance gains from SWaST become more pronounced as coreset size diminishes. This trend aligns with our intuition: smaller coresets present greater selection challenges, where weight pruning facilitates the identification of higher-quality samples (see Fig. 5a and 5c). Simultaneously, reduced coreset sizes exacerbate overfitting, which pruning effectively mitigates (see Fig. 5b).

Additional experiments on noisy datasets corroborate these findings, showing consistent performance improvements across diverse data conditions (see Appendix).



(a) Noise in selected coresets (b) Overfitting (c) Loss differences

Figure 5: Results demonstrating the effectiveness of SWaST (lower values are better in (a) and (b), higher values are better in (c)). (a) Noise ratio in selected coresets across training rounds comparing SWaST to the coreset-only baseline, with a 10.62% reduction in the final selection. (b) Overfitting comparison measured by test loss minus validation loss. SWaST significantly reduces overfitting compared to the dense model (Coreset-only), with higher prune rates yielding progressively better overfitting reduction. (c) Loss ratio between noisy and clean samples during training, showing that weight pruning increases the loss on noisy samples while preserving performance on clean data.



(a) Accuracy vs. pruning rate (b) Accuracy vs. FLOPs

Figure 6: Performance comparison between SWaST-cut and pruning-only baseline. (a) SWaST-cut achieves higher accuracy across different pruning rates, with more pronounced improvements at high sparsity levels. (b) Due to coreset selection, SWaST-cut demonstrates superior accuracy-efficiency trade-offs, significantly outperforming the baseline under equivalent computational budgets.

Coreset selection improves weight pruning. We adopt the coreset algorithm from (Xia et al. 2022) to identify the most informative samples. As shown in Fig. 6a and Fig. 6b, SWaST delivers two key benefits:

- *Higher accuracy*: SWaST achieves 93.15% accuracy at a 90% pruning rate—an improvement of 3.33% over the pruning-only baseline.
- *Better efficiency*: Benefiting from the reduced training overhead through coreset selection, SWaST achieves superior accuracy under comparable FLOPs budgets with a gap of up to 4.43% when the relative FLOPs is 0.01.

Ablation Study

In the following, we conduct comprehensive ablation studies to analyze the key components of our framework.

SWaST improves coreset quality. Extending the experiments in Tab. 1, we further evaluate SWaST by introducing noise into the datasets. Using EL2N (Paul, Ganguli, and Dziugaite 2021) as the coreset selection method (detailed in the appendix), we track the proportion of noisy samples in each selected coreset throughout the training process. The results are visualized as line plots in Fig. 5a. The comparison shows that SWaST-cut helps the coreset algorithm better identify clean samples compared to the coreset-only training baseline. Specifically, in the final coreset, we achieves a 10.62% reduction in the noise ratio relative to the baseline.

SWaST mitigates overfitting on coreset training. Training on small coresets poses inherent overfitting risks due to limited data diversity. We evaluate this phenomenon by comparing test and validation loss between coreset-only training and our SWaST approach. As shown in Fig. 5b, SWaST consistently reduces the overfitting gap across all prune rates, with the reduction effect strengthening progressively as prune rate increases. This demonstrates that our sparse training paradigm effectively regularizes the model by preventing over-specialization to the limited coreset data.

SWaST trains models with superior noise resistance. Following our previous setup, we extend the experiments from Tab. 1 by introducing 10% label noise into the datasets. We analyze the loss distributions of both noisy and clean samples throughout the training process, visualizing the results as line plots in Fig. 5c. The visualization reveals a clear pattern: models with pruning exhibit higher loss values for noisy samples, indicating that pruning prevents the model from memorizing erroneous patterns in noisy data.

Conclusion

We explore the intricate interplay between weight pruning and coreset selection in deep neural networks, uncovering how redundant samples and weights can mutually undermine optimization efficiency and lead to challenges like the double-loss phenomenon. To overcome these issues, we introduce SWaST, an integrated training mechanism that enables simultaneous weight and sample tailoring through iterative compression and a state preservation mechanism. Our extensive experiments reveal a strong synergy between pruning and coreset selection across varying prune rates and coreset sizes, which pave the way for more efficient deep learning paradigms.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (62472097), High-Quality Development Project of Shanghai Municipal Commission of Economy and Informatization (Grant No. 2024-GZL-RGZN-02010), and AI for Science Foundation of Fudan University (FudanX24AI028). The computations in this research were performed using the CFFF platform of Fudan University.

References

Alvarez, J. M.; and Salzmann, M. 2016. Learning the number of neurons in deep networks. *Advances in neural information processing systems*, 29.

Blakeney, C.; Yan, Y.; and Zong, Z. 2020. Is pruning compression?: Investigating pruning via network layer similarity. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 914–922.

Borsos, Z.; Mutny, M.; and Krause, A. 2020. Coresets via bilevel optimization for continual learning and streaming. *Advances in neural information processing systems*, 33: 14879–14890.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Elkerdawy, S.; Elhoushi, M.; Zhang, H.; and Ray, N. 2022. Fire Together Wire Together: A Dynamic Pruning Approach with Self-Supervised Mask Prediction. 2022 IEEE. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)(2022)*, 12444–12453.

Engstrom, L.; Feldmann, A.; and Madry, A. 2024. DsDm: Model-Aware Dataset Selection with Datamodels. *arXiv preprint arXiv:2401.12926*.

Evci, U.; Gale, T.; Menick, J.; Castro, P. S.; and Elsen, E. 2020. Rigging the lottery: Making all tickets winners. In *International conference on machine learning*, 2943–2952. PMLR.

Everaert, D.; and Potts, C. 2023. GIO: Gradient Information Optimization for Training Dataset Selection. *arXiv preprint arXiv:2306.11670*.

Filters'Importance, D. 2016. Pruning Filters for Efficient ConvNets.

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

Grosse, R.; and Martens, J. 2016. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, 573–582. PMLR.

Guan, Y.; Liu, N.; Zhao, P.; Che, Z.; Bian, K.; Wang, Y.; and Tang, J. 2022. Dais: Automatic channel pruning via differentiable annealing indicator search. *IEEE Transactions on Neural Networks and Learning Systems*.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In *ICLR*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Hao, J.; Ji, K.; and Liu, M. 2024. Bilevel Coreset Selection in Continual Learning: A New Formulation and Algorithm. *Advances in Neural Information Processing Systems*, 36.

Hassibi, B.; and Stork, D. 1992. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5.

He, Y.; Kang, G.; Dong, X.; Fu, Y.; and Yang, Y. 2018. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*.

Hoefler, T.; Alistarh, D.; Ben-Nun, T.; Dryden, N.; and Peste, A. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241): 1–124.

Ilyas, A.; Park, S. M.; Engstrom, L.; Leclerc, G.; and Madry, A. 2022. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*.

Jiang, D.; Cao, Y.; and Yang, Q. 2022. On the Channel Pruning using Graph Convolution Network for Convolutional Neural Network Acceleration. In *IJCAI*, 3107–3113.

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

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

Killamsetty, K.; Zhao, X.; Chen, F.; and Iyer, R. 2021c. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *Advances in neural information processing systems*, 34: 14488–14501.

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

Kuhn, H. W.; and Tucker, A. W. 2013. Nonlinear programming. In *Traces and emergence of nonlinear programming*, 247–258. Springer.

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7): 3.

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Lee, N.; Ajanthan, T.; and Torr, P. H. 2018. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

Louizos, C.; Welling, M.; and Kingma, D. P. 2017. Learning sparse neural networks through L_0 regularization. *arXiv preprint arXiv:1712.01312*.

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

Murray, K.; and Chiang, D. 2015. Auto-sizing neural networks: With applications to n-gram language models. *arXiv preprint arXiv:1508.05051*.

Nonnenmacher, M.; Pfeil, T.; Steinwart, I.; and Reeb, D. 2021. Sosp: Efficiently capturing global correlations by second-order structured pruning. *arXiv preprint arXiv:2110.11395*.

Park, D.; Choi, S.; Kim, D.; Song, H.; and Lee, J.-G. 2024. Robust data pruning under label noise via maximizing relabeling accuracy. *Advances in Neural Information Processing Systems*, 36.

Paul, M.; Ganguli, S.; and Dziugaite, G. K. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34: 20596–20607.

Pruthi, G.; Liu, F.; Kale, S.; and Sundararajan, M. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930.

Qian, X.; and Klabjan, D. 2021. A probabilistic approach to neural network pruning. In *International Conference on Machine Learning*, 8640–8649. PMLR.

Renda, A.; Frankle, J.; and Carbin, M. 2020. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*.

Runge, C. 1901. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. *Zeitschrift für Mathematik und Physik*, 46(224-243): 20.

Shibagaki, A.; Karasuyama, M.; Hatano, K.; and Takeuchi, I. 2016. Simultaneous safe screening of features and samples in doubly sparse modeling. In *International Conference on Machine Learning*, 1577–1586. PMLR.

Srinivas, S.; Subramanya, A.; and Venkatesh Babu, R. 2017. Training sparse neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 138–145.

Tanaka, H.; Kunin, D.; Yamins, D. L.; and Ganguli, S. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in neural information processing systems*, 33: 6377–6389.

Thakkar, M.; Bolukbasi, T.; Ganapathy, S.; Vashishth, S.; Chandar, S.; and Talukdar, P. 2023. Self-Influence Guided Data Reweighting for Language Model Pre-training. *arXiv preprint arXiv:2311.00913*.

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

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29.

Xia, M.; Malladi, S.; Gururangan, S.; Arora, S.; and Chen, D. 2024. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*.

Xia, X.; Liu, J.; Yu, J.; Shen, X.; Han, B.; and Liu, T. 2022. Moderate coreset: A universal method of data selection for real-world data-efficient deep learning. In *The Eleventh International Conference on Learning Representations*.

Xiao, W.; Chen, Y.; Shan, Q.; Wang, Y.; and Su, J. 2024. Feature Distribution Matching by Optimal Transport for Effective and Robust Coreset Selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 9196–9204.

Zhang, W.; Hong, B.; Liu, W.; Ye, J.; Cai, D.; He, X.; and Wang, J. 2017. Scaling up sparse support vector machines by simultaneous feature and sample reduction. In *International Conference on Machine Learning*, 4016–4025. PMLR.

Zhou, X.; Pi, R.; Zhang, W.; Lin, Y.; Chen, Z.; and Zhang, T. 2022. Probabilistic bilevel coreset selection. In *International Conference on Machine Learning*, 27287–27302. PMLR.

Zhou, X.; Zhang, W.; Xu, H.; and Zhang, T. 2021. Effective sparsification of neural networks with global sparsity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3599–3608.