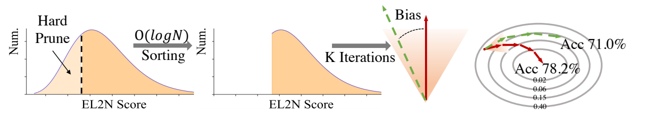
InfoBatch: Lossless Training Speed Up by Unbiased Dynamic Data Pruning

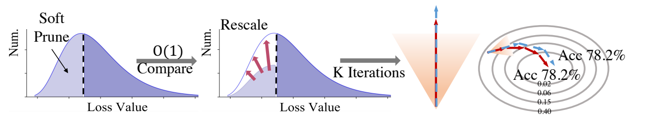
Ziheng Qin[*]   Kai Wang[*†]   Zangwei Zheng   Jianyang Gu
Xiangyu Peng   Daquan Zhou   Yang You[‡]
National University of Singapore
{zihengq, kai.wang, youy}@comp.nus.edu.sg
Code: https://github.com/henryqin1997/InfoBatch

Abstract

Data pruning aims to obtain lossless performances as training on the original data with less overall cost. A common approach is to simply filter out samples that make less contribution to the training. This leads to gradient expectation bias between the pruned and original data. To solve this problem, we propose ***InfoBatch****, a novel framework aiming to achieve lossless training acceleration by unbiased dynamic data pruning. Specifically, InfoBatch randomly prunes a portion of less informative samples based on the loss distribution and rescales the gradients of the remaining samples. We train the full data in the last few epochs to improve the performance of our method, which further reduces the bias of the total update. As a plug-and-play and architecture-agnostic framework, InfoBatch consistently obtains lossless training results on CIFAR-10, CIFAR-100, Tiny-ImageNet, and ImageNet-1K saving 40%, 33%, 30%, and 26% overall cost, respectively. We extend InfoBatch into semantic segmentation task and also achieve lossless mIoU on ADE20K dataset with 20% overall cost saving. Last but not least, as InfoBatch accelerates in data dimension, it further speeds up large-batch training methods (eg. LARS and LAMB) by 1.3 times without extra cost or performance drop. The code will be made public.*



(a) Previous methods [42, 35, 36] prune samples via setting some heuristic metrics (*e.g.* EL2N score). The hard pruning operation results in biased gradient expectation (green lines) and sub-optimal training results.



(b) Our proposed InfoBatch randomly prunes samples with small loss values and rescales the updates of them. Thereby InfoBatch maintains approximately the same gradient expectation (blue lines) as training on the original dataset.

Figure 1: Visualization of difference between InfoBatch and a previous data pruning method (EL2N [35]). InfoBatch achieves lossless acceleration performance compared to previous works, while EL2N encounters a performance drop of 7.2%. Experiments are conducted with ResNet-18 under the same pruning ratio of 30% on the CIFAR-100 dataset. The semi-transparent triangles represent the variance range of gradient estimation.

1. Introduction

In the past decade, deep learning has achieved remarkable progress in the computer vision area [12, 43, 17, 37]. Most state-of-the-art methods [12, 43, 27] are trained on ultra-large-scale datasets, but the heavy training cost is hardly affordable for researchers with limited computing resources. Reducing the training effort for large-scale datasets has become urgent for broader computer vision and other deep-learning applications.

An intuitive solution is to reduce the training sample amount. Dataset distillation [50, 45, 3, 32, 33] and coreset selection [16, 4, 44, 40] respectively synthesize or choose a small but informative dataset from the original large one. Although the sample amount is reduced, the distillation and selection algorithms lead to extra costs. Besides, these two methods are also hard to achieve lossless performance [51, 50, 19, 3, 45]. Another solution is weighted sampling methods [52, 9, 20] that aim to improve the sampling frequency of certain samples. It improves the convergence speed, but their accelerations are sensitive to models and datasets [52]. LARS [48] and LAMB [49] enable a

[*]Equal contribution.
[†]project lead.
[‡]Corresponding author.

super large batch size to improve data parallelism during training to reduce the overall training time. However, more computation units are required and the total training cost is not reduced, which limits the effect under constraint computation resources.

Most recently, a series of works propose to accelerate training by reducing the total training iterations. [42, 35] estimate a score for each sample and accordingly prune less informative samples. However, these methods usually need several trials to estimate more accurate scores, which requires extra overhead even longer than training time (as shown in Tab. 2) for large-scale datasets (*e.g.* ImageNet-1K). To reduce this heavy overhead, [36] dynamically prunes the samples based on scores, *e.g.*, loss values, during the training without trials. Nevertheless, this work still needs to sort the whole dataset by the score in each pruning cycle, which leads to $O(logN)$ (N denotes the number of samples) per sample time complexity on sorting. Therefore, it is impractical to apply this methods [42, 35, 36] on large-scale datasets, such as ImageNet-1K. Meanwhile, directly pruning data may lead to a biased gradient estimation as illustrated in Fig. 1a, which affects the convergence result. This is a crucial factor that limits their performance, especially under a high pruning ratio.

To tackle these issues, we propose InfoBatch, a novel unbiased dynamic data pruning framework based on the idea of maintaining the same total update between training on the pruned and original datasets. We refer to this idea as expectation rescaling for simplicity. Specifically, given a dataset, we maintain a score of each sample with its loss value during forward propagation. We randomly prune a certain portion of small-score (*i.e.* well-learned) samples in each epoch. Different from previous methods where well-learned samples are dropped directly, as shown in Fig. 1b, we scale up the gradient of those remaining small-score samples to keep the approximately same expectation as the original dataset. Compared to previous works [42, 35, 36], the gradient expectation bias in optimization between InfoBatch and standard training is reduced, as illustrated in the right part of Fig. 1a and 1b. In order to further reduce the bias of the total update, we train with the full dataset in the last few epochs. We provide theoretical analysis in Sec 3.3 to prove our proposed InfoBatch is an unbiased dynamic data pruning framework.

InfoBatch is compatible with various deep-learning tasks. In this paper, we investigate its effect on classification and semantic segmentation tasks. Under the same batch size and computational resource budget, InfoBatch achieves lossless training performances with 20% $\sim$ 40% less overall cost across various tasks and architectures. It helps mitigate the heavy computation cost for training on ultra-large-scale datasets. The time complexity of InfoBatch is $O(1)$ per sample, which is significantly faster than previous works ($O(logN)$), especially when N is huge. Our contributions are summarized as follows,

- We propose a novel framework termed as InfoBatch, aiming to achieve lossless training acceleration by unbiased dynamic data pruning.
- InfoBatch prunes less informative samples based on loss value distribution and maintains the same gradient update expectation as training on the original dataset via expectation scaling.
- InfoBatch reduces the overall cost by 20% $\sim$ 40% across various architectures on different datasets including CIFAR-10, CIFAR-100, ImageNet-1K, and ADE20K, showing its effectiveness and generality for accelerating the training of deep-learning.

2. Related Works

Static Data Pruning. The motivation for static data pruning methods [42, 35, 19, 22, 13] is to utilize fewer samples while achieve comparable results as the original dataset. Almost all the methods are based on predefined or heuristic metrics. These metrics can be roughly divided into the following categories: geometry-based [1, 39], uncertainty-based [6], error-based [42, 35], decision-boundary-based [13], gradient-matching [31, 21], bilevel optimization [22] and submodularity-based methods [19]. Contextual Diversity (CD) [1], Herding [46], and k-Center remove the redundant samples based on their similarity to the rest of the data. Cal [30] and Deepfool [13] select samples based on their difficulties for learning. FL [19] and Graph Cut (GC) [19] consider the diversity and information simultaneously by maximizing the submodular function. Recently GraNd and EL2N [35] propose to estimate sample importance with gradient norm and error-L2-norm. The main limitation of these works can be summarized as follows: 1). The predefined or heuristic metrics can not work well across architectures or datasets. 2). The extra cost of these methods is not negligible. As illustrated in Tab. 2, the extra time cost of EL2N is 17.5 hours, which is even longer than the 12.3-hour training time.

Dynamic data pruning. Dynamic data pruning aims to save the training cost by reducing the number of iterations for training. The pruning process is conducted during training and sample information can be obtained from current training. [36] proposes two dynamic pruning methods called UCB and ϵ-greedy. An uncertainty value is defined and the estimated moving average is calculated. Once for every pruning period, ϵ-greedy/UCB is used to select a given fraction of the samples with the highest scores and then trains on these samples during the period. Under this dynamic pruning setting, it achieves a favorable performance compared to static pruning methods on CIFAR-

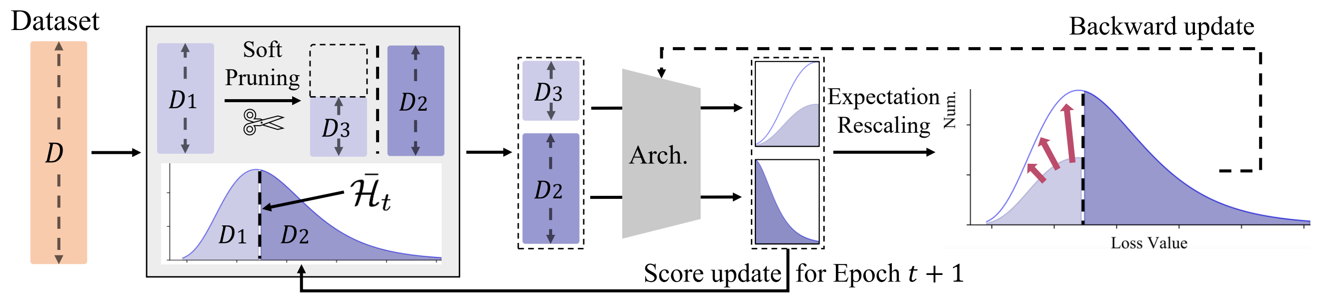


Figure 2: Illustration of the proposed *InfoBatch* framework. InfoBatch mainly consists of two operations, named soft pruning and expectation rescaling. $\bar{\mathcal{H}}_t$ denotes the mean value of scores of samples. Soft pruning randomly prunes some samples from $\mathcal{D}_1$ with small scores. For remaining samples from $\mathcal{D}_1$, expectation rescaling scales up the losses to keep the approximately same gradient expectation as the original dataset.

10/100[25, 26], while saving the overhead of assessing samples and pruning before training. Yet the lossless pruning ratio of a new dataset is still unknown in advance. Besides, the sorting operation (cost $O(logN)$ per sample on dataset size N) for obtaining samples with the highest scores is a huge overhead for extra-large datasets (e.g. ImageNet-1K has 1.28 million pictures) as it is called multiple times.

Other related works. Large batch training fully exploits the accelerator's parallelism power. Nonetheless, it cannot reduce the overall cost. With fewer iterations, a large learning rate is needed [14, 18] to keep the update size but make the training unstable. LARS [48] and LAMB [49] stabilize the training process by normalizing layer-wise gradients. Stochastic optimization with importance sampling[52, 9, 20] tries to accelerate the optimization convergence speed by sampling certain samples more frequently. A drawback is that the speed-up is sensitive to the model and dataset[52]. It cannot directly fit into a training scheme without knowing how much it speeds up.

3. Methods

In this section, we first review the preliminaries of static data pruning (including coreset selection) [35, 42, 7, 19, 22, 21] and dynamic data pruning [36]. Then we introduce InfoBatch and its components. Finally, we provide a detailed theoretical analysis to prove that InfoBatch achieves unbiased data pruning results.

3.1. Preliminaries

In this part, we briefly introduce preliminaries of several data pruning strategies.

Static Pruning. Given a large-scale dataset $\mathcal{D} = \{z_i\}|_{i=1}^{|\mathcal{D}|} = \{(x_i, y_i)\}|_{i=1}^{|\mathcal{D}|}$, we can define a score $\mathcal{H}(z)$ for each sample. During training, samples are drawn from a pruning probability $\mathcal{P}$ defined on top of $\mathcal{H}$. Static pruning aims to select a static subset that achieves comparable results as the original dataset with less storage and training cost. It directly discards all samples satisfying a certain condition. This results in $\mathcal{P}(z;\mathcal{H}) \in \{0, 1\}$. For examples, [42] defines:

$$\mathcal{P}(z;\mathcal{H}) = \mathbb{1}(\mathcal{H}(z) < \bar{\mathcal{H}}_t), \tag{1}$$

where $\bar{\mathcal{H}}_t$ is a threshold and $\mathbb{1}(\cdot)$ is indicator function. A subset $\mathcal{S}$ is formed by pruning samples with $\mathcal{P}(z;\mathcal{H}) = 1$.

Fig. 1a briefly illustrates the whole process of static pruning. One can find that directly dropping the samples with low scores and training on the rest samples could be prone to a gradient expectation bias compared to training on the original dataset. We also provide a more detailed theoretical analysis in Appendix.

Dynamic Pruning. Different from static pruning, dynamic pruning aims to save the training cost by reducing the number of iterations during the training[36]. Also, scoring and selection are both necessary. The biggest difference is that the score $\mathcal{H}_t$ can change along with the training process, where t denotes the temporal status. As a result, the probability is also a step-dependent one:

$$\mathcal{P}_t = \mathcal{P}(z;\mathcal{H}_t), \tag{2}$$

and forms a dynamic pruned dataset $\mathcal{S}_t$.

Compared to static pruning, dynamic pruning has access to all the original data during training. Thus the gradient expectation bias should be much smaller than static pruning. However, such a scheme still has the following limitations: i). As claimed in [36], low-score samples of different t during the training could easily overlap, and directly pruning them every time may still lead to a bias (see it in Fig. 1a). ii). Pruning samples leads to reduced number of gradient updates. Under the premise of saving training cost, many dynamic pruning methods [36] hardly achieve lossless results compared to training on the original dataset. iii). Scoring and sorting operations in dynamic pruning are conducted multiple times, the overhead of which limits the application on large-scale datasets.

3.2. Overview of InfoBatch

Based on the above observation and analysis, we propose InfoBatch, a novel framework for achieving lossless training acceleration based on unbiased dynamic data pruning. As illustrated in Fig. 2, we maintain a score for each sample with its loss value during forward propagation. We set the mean of these values as the pruning threshold. A certain portion of small-score samples is accordingly pruned in each epoch. Then, to obtain the same expectation of gradient as the original dataset in each epoch, we scale up the gradients of the remaining small-score samples. By doing this, compared to previous static and dynamic pruning methods [42, 35, 7, 36, 19], the performance differences between training on the pruned dataset and the original dataset are reduced. In order to further reduce the bias of the total update, we train with the full dataset in the last few epochs.

3.3. Unbiased Prune and Rescale

InfoBatch adopts the dynamic pruning process as in Eqn. 2. We first define our pruning policy $\mathcal{P}_t$. Previous methods using a deterministic pruning operation could cause bias as discussed above. In contrast, we introduce randomness into the pruning process to solve this. Given a dataset $\mathcal{D}$, in t-th epoch, we assign a pruning probability to each sample based on its score. Such a soft pruning policy is formulated as:

$$\mathcal{P}_t(z) = \begin{cases} r, & \mathcal{H}_t(z) < \bar{\mathcal{H}}_t \\ 0, & \mathcal{H}_t(z) \geq \bar{\mathcal{H}}_t \end{cases}, \tag{3}$$

where $\bar{\mathcal{H}}_t$ is the mean value of all the scores $\mathcal{H}_t$ and $r \in (0,1)$ is a predefined hyper-parameter as the pruning probability. Our new prune policy has the following benefits: i). Soft pruning allows each small-score sample to be utilized for training, which reduces the bias caused by hard pruning in previous dynamic pruning methods. ii). Our proposed strategy is based on the comparison with $\bar{\mathcal{H}}_t$, with no requirement to sort the whole training samples, which reduces the time complexity from $O(logN)$ to $O(1)$. It indicates that InfoBatch could be practical on large-scale datasets.

Then, we utilize loss values $\mathcal{L}(z)$ of each sample as the corresponding score based on the following two reasons: i). loss values can be obtained without extra cost, ii). loss values reflect the learning status of samples [5]. Specifically, in the t-th ($t > 0$) epoch, we utilize the soft pruning policy to prune samples based on their scores. After that, for the pruned samples, their scores remain unmodified as previous. For the remaining samples, their scores are updated by the losses in the current epoch. Mathematically, $\mathcal{H}_t(z)$ would be updated by the latest losses to $\mathcal{H}_{t+1}(z)$ for the following epochs:

$$\mathcal{H}_{t+1}(z) = \begin{cases} \mathcal{H}_t(z), & z \in \mathcal{D} \backslash \mathcal{S}_t \\ \mathcal{L}(z), & z \in \mathcal{S}_t \end{cases}. \tag{4}$$

Note that, for the first epoch, we initialize the scores with $\{1\}$ provided no previous loss.

There are several significant benefits of our soft pruning policy, yet it still cannot avoid the influence caused by the less number of gradient updates. To address this issue, we scale up the gradients of the remaining samples. Specifically, given a remaining sample with score $\mathcal{H}(z) < \bar{\mathcal{H}}_t$, whose corresponding pruning probability is r, we rescale its gradient to $1/(1-r)$. For the samples with scores larger than $\bar{\mathcal{H}}_t$, the loss is not modified. Thereby the gradient update expectation is approximately equal to training on the original dataset. Besides, as the rescaling is operated on small-score samples, it further refines the direction of gradient update expectation. We provide the following theoretical analysis to demonstrate the necessity and advantages of the expectation rescaling operation.

Theoretical Analysis. We can interpret the training objective as minimizing empirical risk $\mathcal{L}$. Assuming all samples z from $\mathcal{D}$ are drawn from continuous i.i.d. distribution $\rho(z)$, we can establish the training objective as:

$$\underset{\theta \in \Theta}{\arg\min} \underset{z \in \mathcal{D}}{\mathbb{E}} [\mathcal{L}(z,\theta)] = \int_z \mathcal{L}(z,\theta)\rho(z)dz. \tag{5}$$

After applying our proposed pruning, we sample z according to normalized $(1-\mathcal{P}_t(z))\rho(z)$. In backpropagation, rescaling loss is equivalent to rescaling the gradient. By rescaling the loss of each sample z with a factor $\gamma_t(z)$ ($\forall z \in \mathcal{D}, \mathcal{P}_t(z) = 0 \Rightarrow \gamma_t(z) = 1$), the training objective on $\mathcal{S}_t$ becomes:

$$\begin{aligned} &\underset{\theta \in \Theta}{\arg\min} \underset{z \in \mathcal{S}_t}{\mathbb{E}} [\gamma_t(z)\mathcal{L}(z,\theta)] \\ =& \underset{\theta \in \Theta}{\arg\min} \frac{\int_z (1-\mathcal{P}_t(z))\gamma_t(z)\mathcal{L}(z,\theta)\rho(z)dz}{\int_z (1-\mathcal{P}_t(z))\rho(z)dz}. \end{aligned} \tag{6}$$

By setting $\gamma_t(z) = 1/(1-\mathcal{P}_t(z))$, Eqn. 6 becomes

$$\underset{\theta \in \Theta}{\arg\min} \frac{1}{c_t} \int_z \mathcal{L}(z,\theta)\rho(z)dz, \tag{7}$$

where $c_t = \mathbb{E}_{z\sim\rho}[1-\mathcal{P}_t(z)] = \int_z \rho(z)(1-\mathcal{P}_t(z))dz$, $c_t \in (0,1)$ is a constant for temporal status t. Then the objective in Eqn. 7 is a constant-rescaled version of the original objective in Eqn. 5. Therefore, training on $\mathcal{S}_t$ with rescaled factor $\gamma_t(z)$ could achieve a similar result as training on the original dataset. Furthermore, we find these operations also leverage the problem of reduced iterations. In

Table 1: The accuracy (%) comparison to state-of-the-art methods. All methods are trained with ResNet-18. As InfoBatch has a self-adaptive ratio, we mark the results with † where the same forward propagation number during training are matched. Random* denotes dynamic random pruning. Due to the differences of hyper-parameter designs, for static methods, we report the Tiny-ImageNet results in the original paper. Our results are reproduced by ourselves. Details are available in Appendix.

	Dataset	CIFAR10 [25]			CIFAR100 [26]			Tiny-ImageNet [27]			
	Prune Ratio %	70	50	30	70	50	30	50	40	30	0
Static	Random	$90.2_{\downarrow 5.4}$	$93.3_{\downarrow 2.3}$	$94.6_{\downarrow 1.0}$	$69.7_{\downarrow 8.5}$	$72.1_{\downarrow 6.1}$	$73.8_{\downarrow 4.4}$	$38.2_{\downarrow 12.9}$	$41.2_{\downarrow 9.9}$	$44.3_{\downarrow 6.8}$	$51.1_{\pm 0.2}$
	CD [1]	$90.8_{\downarrow 4.8}$	$94.3_{\downarrow 1.3}$	$95.0_{\downarrow 0.6}$	$70.3_{\downarrow 7.9}$	$72.3_{\downarrow 5.9}$	$74.2_{\downarrow 4.0}$	-	-	-	-
	Herding [46]	$80.1_{\downarrow 15.5}$	$88.0_{\downarrow 7.6}$	$92.2_{\downarrow 3.4}$	$69.6_{\downarrow 8.0}$	$71.8_{\downarrow 6.4}$	$73.1_{\downarrow 5.1}$	$38.8_{\downarrow 12.3}$	$42.2_{\downarrow 8.9}$	$44.8_{\downarrow 6.3}$	$51.1_{\pm 0.2}$
	K-Center [39]	$90.9_{\downarrow 4.7}$	$93.9_{\downarrow 1.7}$	$94.7_{\downarrow 0.9}$	$70.2_{\downarrow 8.0}$	$72.2_{\downarrow 6.0}$	$74.1_{\downarrow 4.1}$	-	-	-	-
	Least Confidence [6]	$90.3_{\downarrow 5.3}$	$94.5_{\downarrow 1.1}$	$95.0_{\downarrow 0.6}$	$69.8_{\downarrow 8.4}$	$72.3_{\downarrow 5.9}$	$74.2_{\downarrow 4.0}$	-	-	-	-
	Margin [6]	$90.9_{\downarrow 4.7}$	$94.3_{\downarrow 1.3}$	$94.9_{\downarrow 0.7}$	$70.2_{\downarrow 8.0}$	$72.2_{\downarrow 6.0}$	$74.0_{\downarrow 4.2}$	-	-	-	-
	Forgetting [42]	$91.7_{\downarrow 3.9}$	$94.1_{\downarrow 1.5}$	$94.7_{\downarrow 0.9}$	$69.9_{\downarrow 8.3}$	$73.1_{\downarrow 5.1}$	$75.3_{\downarrow 2.9}$	$39.1_{\downarrow 12.0}$	$42.5_{\downarrow 8.6}$	$46.3_{\downarrow 4.8}$	$51.1_{\pm 0.2}$
	GraNd [35]	$91.2_{\downarrow 4.4}$	$94.6_{\downarrow 1.0}$	$95.3_{\downarrow 0.3}$	$68.8_{\downarrow 9.4}$	$71.4_{\downarrow 6.8}$	$74.6_{\downarrow 3.6}$	$37.2_{\downarrow 13.9}$	$40.8_{\downarrow 10.3}$	$45.1_{\downarrow 6.0}$	$51.1_{\pm 0.2}$
	DeepFool [13]	$90.0_{\downarrow 5.6}$	$94.1_{\downarrow 1.5}$	$95.1_{\downarrow 0.5}$	$69.8_{\downarrow 6.4}$	$73.2_{\downarrow 5.0}$	$74.2_{\downarrow 4.0}$	-	-	-	-
	Craig [31]	$88.4_{\downarrow 7.2}$	$93.3_{\downarrow 3.3}$	$94.8_{\downarrow 0.8}$	$69.7_{\downarrow 8.5}$	$71.9_{\downarrow 6.3}$	$74.4_{\downarrow 3.8}$	-	-	-	-
	Glister [22]	$90.9_{\downarrow 4.7}$	$94.0_{\downarrow 1.6}$	$95.2_{\downarrow 0.4}$	$70.4_{\downarrow 7.8}$	$73.2_{\downarrow 5.0}$	$74.6_{\downarrow 3.6}$	-	-	-	-
	Influence [24]	$88.3_{\downarrow 7.3}$	$91.3_{\downarrow 4.3}$	$93.1_{\downarrow 2.5}$	$68.9_{\downarrow 9.5}$	$72.0_{\downarrow 6.2}$	$74.4_{\downarrow 3.8}$	$37.7_{\downarrow 13.4}$	$41.2_{\downarrow 9.9}$	$45.3_{\downarrow 5.8}$	$51.1_{\pm 0.2}$
	EL2N [42]	$89.8_{\downarrow 5.8}$	$93.2_{\downarrow 2.4}$	$94.4_{\downarrow 1.2}$	$68.5_{\downarrow 9.7}$	$71.0_{\downarrow 7.2}$	$74.1_{\downarrow 4.1}$	$37.1_{\downarrow 14.0}$	$40.1_{\downarrow 11.0}$	$44.5_{\downarrow 6.6}$	$51.1_{\pm 0.2}$
	DP [47]	$90.8_{\downarrow 4.8}$	$93.8_{\downarrow 1.8}$	$94.9_{\downarrow 0.7}$	-	$73.1_{\downarrow 5.1}$	$77.2_{\downarrow 1.0}$	$40.5_{\downarrow 10.6}$	$43.6_{\downarrow 7.5}$	$46.7_{\downarrow 4.4}$	$51.1_{\pm 0.2}$
Dynamic	Random*	$93.0_{\downarrow 2.6}$	$94.5_{\downarrow 1.1}$	$94.8_{\downarrow 0.8}$	-	$75.3_{\downarrow 2.9}$	$77.3_{\downarrow 0.9}$	-	-	-	-
	ϵ-greedy[36]	$94.1_{\downarrow 1.5}$	$94.9_{\downarrow 0.7}$	$95.2_{\downarrow 0.4}$	-	$74.8_{\downarrow 3.4}$	$76.4_{\downarrow 1.8}$	-	-	-	-
	UCB [36]	$93.9_{\downarrow 1.7}$	$94.7_{\downarrow 0.9}$	$95.3_{\downarrow 0.3}$	-	$75.3_{\downarrow 2.9}$	$77.3_{\downarrow 0.9}$	-	-	-	-
	InfoBatch	$^{\dagger}\mathbf{94.7}_{\downarrow 0.9}$	$^{\dagger}\mathbf{95.1}_{\downarrow 0.5}$	$\mathbf{95.6}_{\uparrow 0.0}$	$^{\dagger}\mathbf{77.6}_{\downarrow 0.6}$	$^{\dagger}\mathbf{78.2}_{\uparrow 0.0}$	$\mathbf{78.8}_{\uparrow 0.6}$	$\mathbf{59.5}_{\uparrow 0.0}$	$\mathbf{59.6}_{\uparrow 0.1}$	$\mathbf{60.2}_{\uparrow 0.7}$	$59.5_{\pm 0.4}$
	Whole Dataset	$95.6_{\pm 0.1}$			$78.2_{\pm 0.1}$			$59.5_{\pm 0.4}$			

real-world application, we consider the dataset as discrete case and provide specific analysis as follows,

$$\mathbb{E}\left[\frac{1}{c_t}\right] = \frac{|\mathcal{D}|}{\sum_{z\in\mathcal{D}}(1-\mathcal{P}_t(z))} \simeq \frac{|\mathcal{D}|}{|\mathcal{S}_t|}. \tag{8}$$

In implementation we force a deterministic result of $\frac{1}{c_t} = \frac{|\mathcal{D}|}{|\mathcal{S}_t|}$, so that we can substitute Eqn. 8 into Eqn. 7, and do a differentiation of loss $\mathcal{L}$ over θ as follows:

$$\mathbb{E}[\nabla_\theta \mathcal{L}(\mathcal{S}_t)] \simeq \frac{|\mathcal{D}|}{|\mathcal{S}_t|}\,\mathbb{E}[\nabla_\theta \mathcal{L}(\mathcal{D})]. \tag{9}$$

For each epoch, the iteration number, *i.e.* gradient update number becomes $\frac{|\mathcal{S}_t|}{|\mathcal{D}|}$ of the original one, while our method scale the expected gradient to $\frac{|\mathcal{D}|}{|\mathcal{S}_t|}$. As a result, this would leverage the influence of reduced gradient update number. The approximation will hold when the pruning ratio is not too high. We provide a detailed analysis in the Appendix.

3.4. Annealing

Based on the theoretical analysis above, the objectives and updating expectations between InfoBatch and training on the original dataset are approximately the same. However, there still exist minor differences between the optimization on $\mathcal{D}$ and $\mathcal{S}_t$. During training, if a sample is pruned in the middle stage, it is still likely to be revisited afterwards. However, in the last few epochs, the revisiting probability drastically drops, resulting in remaining gradient expectation bias. Therefore, given training epoch C, we define a ratio hyper-parameter $\delta \in (0, 1)$. The pruning is only conducted in the first $\delta \cdot C$ epochs. After that, we train on the full dataset till end. The corresponding operation can be interpreted as

$$\mathcal{P}_t(z) = \begin{cases} r, & \mathcal{H}_t(z) < \bar{\mathcal{H}}_t \wedge t < \delta \cdot C \\ 0, & \mathcal{H}_t(z) \geq \bar{\mathcal{H}}_t \vee t \geq \delta \cdot C \end{cases}. \tag{10}$$

Combining the above components, InfoBatch achieves lossless training performance with fewer iterations compared with training on the original dataset.

4. Experiments

4.1. Datasets and Implementation Details

We verify the effectiveness of our method on multiple popular datasets, including CIFAR-10/100 [25, 26], ImageNet-1K [11] and ADE20K [53].

CIFAR-10/100. Two CIFAR datasets consist of 32 × 32 size colored natural images divided into 10 and 100 categories, respectively. In each dataset, 50,000 images are used for training and 10,000 images for testing.

ImageNet-1K is the subset of the ImageNet-21k dataset with 1,000 categories. It contains 1,281,167 training images and 50,000 validation images.

ADE20K is a popular semantic segmentation benchmark. It contains more than 20,000 images with pixel-level annotations of 150 semantic categories.

Implementation Details. All experiments are conducted with an A100 GPU server. For InfoBatch, default value $r = 0.5$ and $\delta = 0.875$ are used if not specified. For classification tasks, We train ResNet18 and ResNet-50[17] for evaluation. On CIFAR-10/100 and ImageNet-1K, all models are trained with OneCycle scheduler (with cosine annealing) [41, 29] with default setting and LARS optimizer [48] with momentum 0.9, weight decay 5e-4; On Tiny-ImageNet, we use SGD [38] optimizer and cosine annealing optimizer. All images are augmented with commonly adopted transformations, *i.e.* normalization, random crop, and horizontal flop. The implementation is based on PyTorch [34]. For the semantic segmentation task, we conduct experiments on ADE20K [53]. The chosen network is UperNet with backbone R-50. We follow the default configuration of the mmsegmentation [8]. All other details can be found in Appendix. The code is available in the supplementary material.

4.2. Comparisons with SOTA methods

Performance Comparisons. We compare our proposed InfoBatch with static and dynamic data pruning methods in Tab. 1 on CIFAR-10 [25], CIFAR-100 [26], and Tiny-ImageNet [27] datasets. In the first part, we introduce static pruning methods, the simplest baseline of which is random selection before training. Influence [24] and EL2N [35] are two classical static pruning methods that prune samples based on Influence-score and EL2N-score. DP [47] conducts pruning with consideration of generalization. To make a wider comparison, we include 10 coreset selection methods. These methods [19, 1, 46, 39, 6, 35, 13, 31, 22, 24] select a coreset of data via their predefined score function or heuristic knowledge. Then, we introduce 3 dynamic pruning methods in the second part. Following [36], we also construct a dynamic pruning baseline, termed as Random*, which conducts random selection in each epoch. Compared to Random operation in Tab. 1, the diversity of training samples in Random* is much better than Random. Therefore, as shown in Tab. 1, Random* usually performs better than Random. ϵ-greedy [36] is inspired by reinforcement learning. UCB [36] proposes to prune samples using the upper confidence bound of loss.

Based on the comparisons in Tab. 1, we have the following observations: 1). Under 30% pruning ratio, only InfoBatch achieves lossless performances among the three datasets. Other methods only obtain near-lossless results on CIFAR-10 dataset while encountering large performance drops on CIFAR-100 and Tiny-ImagNet datasets. 2). As the pruning ratio increases, InfoBatch continuously outperforms other methods by an even larger accuracy margin.

Table 2: Comparison of performance and time cost on ImageNet-1K. Results are reported with ResNet-50 under 30% prune ratio for 90 epochs on an 8-A100-GPU server. The extra cost denotes overhead caused by the used algorithm. All the results are obtained from the same hardware.

	GC	EL2N	UCB	Ours	Whole Dataset
Performance (%)	$74.5_{\pm 0.4}$	-	-	**76.6**$_{\pm 0.2}$	$76.4_{\pm 0.2}$
Training time (h)	12.3	12.3	12.3	12.3	17.5
Extra time cost (h)	>24	>17.5	0.04	**0.002**	0.0
Total GPU hour (h)	>122.4	>238.4	98.7	**98.4**	140.0

Table 3: Ablation of proposed operations in the proposed framework. We set $r = 0.5$ in the experiments. Random* is consistent with the definition in Tab. 1.

	Operation		Acc.	
$\mathcal{P}$	Rescaling	Annealing	R-18	R-50
Random*			77.3	79.7
Soft Pruning			77.5	79.9
Soft Pruning	✓		**78.8**	80.1
Soft Pruning		✓	77.8	80.0
Soft Pruning	✓	✓	**78.8**	**80.6**
Full Dataset			78.2	80.6

It validates the effectiveness of the proposed unbiased dynamic pruning strategy.

Efficiency Comparisons. In addition to the performance comparison, we also compare the efficiency between Infobatch and other methods. Although the motivations of these methods are diverse, their main goal is to save training costs. Thus, we report training time, extra cost, and total GPU hours of these methods in Tab. 2. Under the same computational condition, static pruning methods GC [19] and EL2N [42] require even more time than the actual training run to process the pruning. Previous state-of-the-art dynamic data pruning method UCB [36] shows far better efficiency in the pruning process. However, as shown in Tab. 1, UCB is hard to achieve lossless training performance even on CIFAR-100. In contrast, InfoBatch further reduces the extra time cost by 20 times than UCB, taking only 8.4 seconds for 90 epochs of pruning. More importantly, InfoBatch achieves lossless performance on ImageNet-1K at the pruning ratio of 30%.

4.3. Ablation Experiments

We perform extensive ablation experiments to illustrate the effects of InfoBatch. If not stated, the experiments are conducted on CIFAR-100 by default.

Evaluating the components of InfoBatch. We design an ablation study to investigate the soft pruning policy, expectation rescaling, and annealing operations in the In-

Table 4: Evaluation of different prune conditions. The samples satisfying the conditions are pruned under $r = 0.5$. Experiments are conducted on CIFAR-100 using R-50.

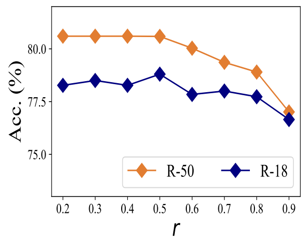
Prune Condition	Acc. (%)	Pruned (%)	GPU Hours (h)
$\mathcal{H}_t(z) < \bar{\mathcal{H}}_t$	80.6	33	1.48
$\mathcal{H}_t(z) > \bar{\mathcal{H}}_t$	80.5	16	1.87
Full Dataset	80.6	0	2.20

foBatch in Tab. 3. Dynamic random pruning, serving as the baseline method, fails to achieve lossless performance on all the architectures. It can be explained that due to pruning, the gradient update number is reduced compared to training on the original dataset. Only applying soft pruning obtain marginally better results than dynamic random pruning. Second, the proposed rescale and anneal operations are consistently complementary and achieve lossless performances in all settings, which aligns with our theoretical analysis. Third, rescaling obtains better correction to the gradient expectation bias on these two tasks than annealing.

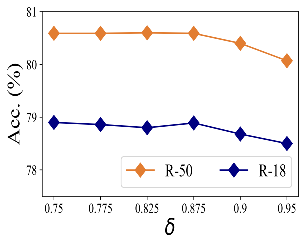
Exploring where to prune. In our default setting, we prune samples with $\mathcal{H}_t(z) < \bar{\mathcal{H}}_t$. Another possible rule is to prune samples with $\mathcal{H}_t(z) > \bar{\mathcal{H}}_t$. We compare the performance and overall training GPU hours of these two prune strategies in Tab. 4. Compared to training on the original dataset, both strategies achieve lossless performance. It demonstrates that InfoBatch is not sensitive to the pruning strategy differences. Compared in more detail, pruning lower score samples is more efficient as it prunes 17% more samples than pruning high score samples. It can be explained by the loss distribution of entropy-based optimization usually being long-tailed [45, 15] as training processes. We provide more visualizations of the loss distribution in the Appendix.

Evaluation of ratio r. We define r as the probability to prune a sample x when $\mathcal{H}_t(z) < \bar{\mathcal{H}}_t$. In Fig. 3a, we evaluate the effect under different pruning ratios with ResNet-18 (R-18) and ResNet-50 (R-50) on CIFAR-100. Setting $r \leq 0.5$ obtains lossless performance on both architectures, which indicates InfoBatch is not too sensitive on hyper-parameter r. Another finding is that setting too large $r > 0.5$ leads to degraded performances on both architectures. This is mainly due to the increased variance when too many samples are pruned. Considering the efficiency and performance, we set $r = 0.5$ by default. More discussion and analysis can be found in the Appendix.

Evaluation of δ. To further reduce the gradient expectation bias, we introduced an annealing operation in the last few epochs. δ is defined as a quantile of given training epoch number C. We evaluate it from 0.75 to 0.95 and report the performances in Fig. 3b. A larger δ means fewer annealing epochs, leaving more remaining bias, which results in degraded performance. When $\delta \leq 0.875$, lossless performance can be achieved with the largest overall cost saving. Not many tuning efforts are required for obtaining lossless results.



(a) Evaluation of r.



(b) Evaluation of δ.

Figure 3: Evaluation curves of hyper-parameter r and δ on R-18 and R-50. Best viewed in color.

Table 5: Cross-architecture robustness results of InfoBatch. 'Full Dataset' denotes training on the original dataset.

	CIFAR-10		CIFAR-100		ImageNet-1K	
	R-18	R-50	R-18	R-50	R-18	R-50
Full Dataset	95.6	95.6	78.2	80.6	70.5	76.4
InfoBatch	95.5	95.6	78.8	80.6	70.4	76.8
Overall Saving (%)	39.2	39.3	33.8	33.9	26.1	26.2

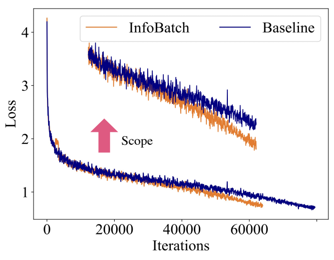
Table 6: Comparison of performance and saved overall cost on CIFAR-10 when trained with R-50 using different optimizers. All the results are obtained from the same hardware.

	SGD	Adam	RAdam	LARS	LAMB
Full Dataset (%)	95.6	94.3	95.0	95.5	95.0
InfoBatch (%)	95.6	94.5	95.0	95.5	95.0
Saved (%)	39.1	37.5	37.4	38.3	39.0

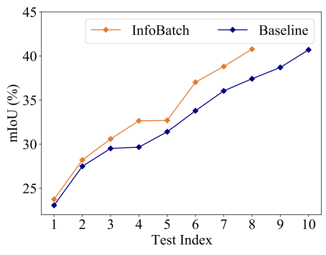
4.4. Generalization Evaluation

Cross-architecture robustness evaluation. InfoBatch is not correlated to specific model architectures, thereby it is a model-agnostic framework. In order to evaluate the cross-architecture generality of InfoBatch, we train R-18 and R-50 on CIFAR-10, CIFAR-100, and ImageNet-1K. As illustrated in Tab. 5, we achieve lossless training performances under all settings, which indicates strong generality of InfoBatch. We also report the overall cost saving in Tab. 5. One can observe that InfoBatch saves more than 26% overall cost on training these datasets. Another finding is that InfoBatch reduces training costs more significantly on easier datasets, such as saving nearly 40% on CIFAR-10.

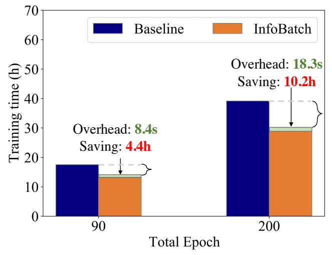
Cross-optimizer robustness evaluation. In the deep-learning area, there are various optimizers [38, 23, 28, 48,



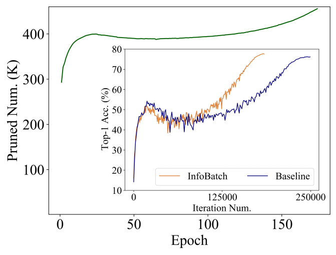
(a) Loss curve on ADE20K.



(b) mIoU curve on ADE20K.



(c) Overhead and savings.



(d) Pruned numbers and Acc.

Figure 4: (a)(b) Evaluation of the proposed InfoBatch framework on ADE20K. With 80% iteration and overall cost, InfoBatch achieves lossless performance on semantic segmentation task. (c) InfoBatch saves the overall cost by more than 25% on ImageNet-1K. (d) The pruned number and accuracy curve during the ImageNet-1K training process. Best viewed in color.

49] that are broadly adopted. We verify InfoBatch's robustness across these optimizers. As shown in Tab. 6, InfoBatch achieves lossless performance with five popular optimizers, and the overall cost saving is stable across optimizers. Specifically, we apply InfoBatch to large-batch training optimizers LARS and LAMB. As InfoBatch accelerates in data dimension, InfoBatch further speeds up training by 1.3 times without extra cost or performance drop. It shows that InfoBatch has the potential to be combined with acceleration methods in other dimensions.

Cross-task robustness evaluation. To verify the generality of InfoBatch on different tasks other than classification, we apply InfoBatch to the semantic segmentation task. Our implementation is based on mmsegmentation [8]. We conduct experiments on ADE20K [53] and report the loss and mIoU curves of training in Fig. 4a and 4b, respectively. InfoBatch uses 80% of original iterations to achieve lossless performance. The result indicates InfoBatch's generality and robustness across different tasks.

4.5. Visualizations

In order to more intuitively demonstrate the effectiveness of InfoBatch, we present two visualizations in this section.

Visualization of savings on ImageNet-1K. To further investigate the efficiency of our InfoBatch in different training epoch settings, we conduct experiments on ImageNet-1K using ResNet-50. We report the 90-epoch and 200-epoch training time comparison of baseline and InfoBatch in Fig. 4c. All the results are obtained with an 8-A100-GPU server. We find that InfoBatch consistently reduces the training time by more than 25%. Furthermore, we show the overhead time cost of InfoBatch in both settings. Compared to the saved hours (4.4h and 10.2h), the cost of InfoBatch (8.4s and 18.3s respectively) is negligible. To explore the characteristic of InfoBatch during training, we visualize the per epoch pruned number and the validation accuracy curve in Fig. 4d. We find that the pruned number steadily increases during early epochs and then keeps stable. This explains why the overall cost saving of our InfoBatch is stable across the training epoch settings as in Fig. 4c. The efficiency and stability are crucial factors for broader applications on ultra-large-scale datasets.

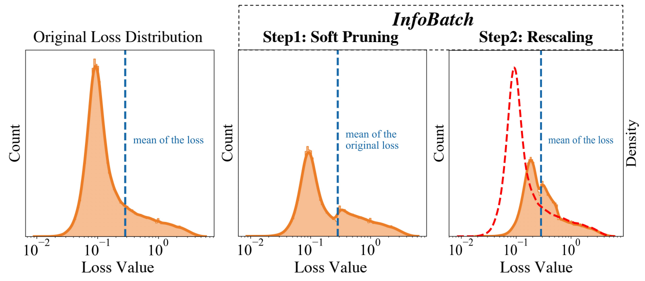


Figure 5: Loss distribution visualizations before and after applying InfoBatch. Best viewed in color.

Visualization of effects on loss distribution. To better illustrate the effects of InfoBatch, we show the loss distribution of CIFAR-10 trained with ResNet50 in the first sub-figure of Fig. 5. Soft pruning randomly prunes samples with loss smaller than the mean as in the second sub-figure in Fig. 5. This leads to insufficient and biased updates, resulting in performance degradation. The rescaling operation of InfoBatch shifts the distribution rightwards. In classification tasks, the Cross-Entropy loss is actually the sum of negative log-likelihood. Therefore, rescaling a sample's loss by $1/(r-1)$ times is basically equivalent to duplicating it into $1/(r-1)$ copies. By rescaling, we achieves a loss distribution equivalent to the original one in the first sub-figure. We show the equivalent distribution (the red dashed line) and the actual scaled distribution in the third sub-figure of Fig.5. More detailed analysis will be in Appendix.

5. Conclusion

We present ***InfoBatch***, a novel framework for lossless training acceleration by unbiased dynamic data pruning. InfoBatch shows its strong robustness on various tasks and datasets, achieving lossless training acceleration on CIFAR-10/100, ImageNet-1K, and ADE20K. InfoBatch reduces the extra overhead cost by at least 20 times compared to previous state-of-the-art methods, which is practical for real-world applications. We provide sufficient experiments and theoretical analysis in this paper and hope it can help the following research in this area.

Limitations and future works. The current version of InfoBatch relies on multi-epoch training schemes. However, GPT-3 [2] and ViT-22B [10] usually train with limited epochs to avoid remembering the knowledge from the training dataset. InfoBatch may not work on these tasks. We are going to explore new strategies for training with [2, 10] in the future.

References

[1] Sharat Agarwal, Himanshu Arora, Saket Anand, and Chetan Arora. Contextual diversity for active learning. In *ECCV*, pages 137–153. Springer, 2020. 2, 5, 6

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 9

[3] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *CVPR*, 2022. 1

[4] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009. 1

[5] Mirza Cilimkovic. Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, 15(1), 2015. 4

[6] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *ICLR*, 2019. 2, 5, 6

[7] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning, 2019. 3, 4

[8] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 6, 8

[9] Dominik Csiba and Peter Richtárik. Importance sampling for minibatches, 2016. 1, 3

[10] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. *arXiv preprint arXiv:2302.05442*, 2023. 9

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. 1

[13] Melanie Ducoffe and Frederic Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*, 2018. 2, 5, 6

[14] 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

[15] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R. Scott, and Dinglong Huang. Curriculumnet: Weakly supervised learning from large-scale web images, 2018. 7

[16] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th annual ACM symposium on Theory of computing*, pages 291–300, 2004. 1

[17] 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. 1, 6

[18] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30, 2017. 3

[19] Rishabh Iyer, Ninad Khargoankar, Jeff Bilmes, and Himanshu Asanani. Submodular combinatorial information measures with applications in machine learning. In *Algorithmic Learning Theory*, pages 722–754. PMLR, 2021. 1, 2, 3, 4, 6

[20] Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 1, 3

[21] 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 *ICML*, pages 5464–5474, 2021. 2, 3

[22] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2, 3, 5, 6

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. 8

[24] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017. 5, 6

[25] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 3, 5, 6

[26] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). 3, 5, 6

[27] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. 1, 5, 6

[28] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2019. 8

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2016. 6

[30] Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. Active learning by acquiring contrastive examples. *arXiv preprint arXiv:2109.03764*, 2021. 2

[31] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *ICML*. PMLR, 2020. 2, 5, 6

[32] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050*, 2020. 1

[33] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *NeurIPS*, 34:5186–5198, 2021. 1

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[35] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training, 2021. 1, 2, 3, 4, 5, 6

[36] Ravi S Raju, Kyle Daruwalla, and Mikko Lipasti. Accelerating deep learning with dynamic data pruning, 2021. 1, 2, 3, 4, 5, 6

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1

[38] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 6, 8

[39] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018. 2, 5, 6

[40] Jae-hun Shim, Kyeongbo Kong, and Suk-Ju Kang. Core-set sampling for efficient neural architecture search. *arXiv preprint arXiv:2107.06869*, 2021. 1

[41] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2017. 6

[42] 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, 2018. 1, 2, 3, 4, 5, 6

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2020. 1

[44] Morad Tukan, Alaa Maalouf, and Dan Feldman. Coresets for near-convex functions. *NeurIPS*, 33, 2020. 1

[45] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *CVPR*, pages 12196–12205, 2022. 1, 7

[46] Max Welling. Herding dynamical weights to learn. In *ICMLg*, pages 1121–1128, 2009. 2, 5, 6

[47] Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence. In *The Eleventh International Conference on Learning Representations*, 2023. 5, 6

[48] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks, 2017. 1, 3, 6, 8

[49] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2019. 1, 3, 8

[50] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. *arXiv*, 1(2):3, 2021. 1

[51] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *ICLR*, 1(2):3, 2021. 1

[52] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling, 2014. 1, 3

[53] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 5, 6, 8