RECYCLE THEN DROP: SELECTIVE RECYCLING FOR EFFICIENT DATA PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training larger models on more data has improved performance but also increased training cost. Dynamic data pruning reduces this cost by scoring each sample with the current model and excluding those the model already handles easily. These pruning scores are typically computed from augmented inputs. However, a score under one augmentation does not establish that the sample has little training value under another. Direct exclusion can therefore leave useful training signal unused. To address this, we propose Recycle Then Drop (RtD), which selects a fraction of the exclusion candidates and trains them with stronger augmentation. This extends the binary pruning decision to keeping, recycling, or dropping samples. Because these decisions still require scoring every sample, we further introduce an amortized variant, A-RtD, which periodically updates and reuses them to reduce repeated scoring and input processing. Experiments across five backbones, multiple datasets, and various augmentation settings demonstrate improvements in accuracy and training efficiency. On CIFAR, RtD reduces the number of samples used for backpropagation by over 80%, while A-RtD reduces training time by up to 54%, both achieving higher accuracy than full data training. Both methods also outperform full data training on object detection and semantic segmentation while requiring less backpropagation. Additional evaluations demonstrate compatibility with existing pruning methods. These results establish selective recycling as a practical approach to improving the accuracy and efficiency of data pruning.

1 INTRODUCTION

Scaling up models and training data has driven progress in deep learning (Sorscher et al., 2022), but has also increased training costs (Mindermann et al., 2022). These costs make such models difficult to train with limited resources, motivating efforts to maintain accuracy while reducing training computation (Qin et al., 2023). Data pruning (Sorscher et al., 2022; He et al., 2024) addresses this goal by reducing the number of training samples. Static pruning (Paul et al., 2021; He et al., 2024) selects a subset before training and uses it throughout training. However, models learn some samples quickly while others require more training (Yuan et al., 2025), so the samples that need further training change as learning progresses. Dynamic data pruning responds to these changes by updating sample selection using various criteria. Examples include training loss (Qin et al., 2023), its change over time (Yuan et al., 2025), and prediction confidence (Sathiyanarayanan et al., 2025). These criteria decide which samples to exclude from each update, reducing repeated backpropagation.

The scores used for these decisions, however, are obtained from the inputs presented to the model. Changing the augmentation can change both the input and the training signal obtained from the same sample (Hu et al., 2021). For example, confidence-based pruning (Sathiyanarayanan et al., 2025) can mark a sample for exclusion when its augmented input meets a confidence threshold. In this case, high confidence is observed under the augmentation used for scoring and does not establish that the sample has little training value under other augmentations. In other words, augmentation-level redundancy does not imply sample-level redundancy. Yet training all exclusion candidates again would sacrifice the savings from skipping their updates. We therefore need a way to learn from some exclusion candidates while retaining the savings from dropping the rest. We propose Recycle Then Drop (RtD), which extends binary pruning by recycling a fraction of the exclusion candidates under stronger augmentation and dropping the rest from the current update.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

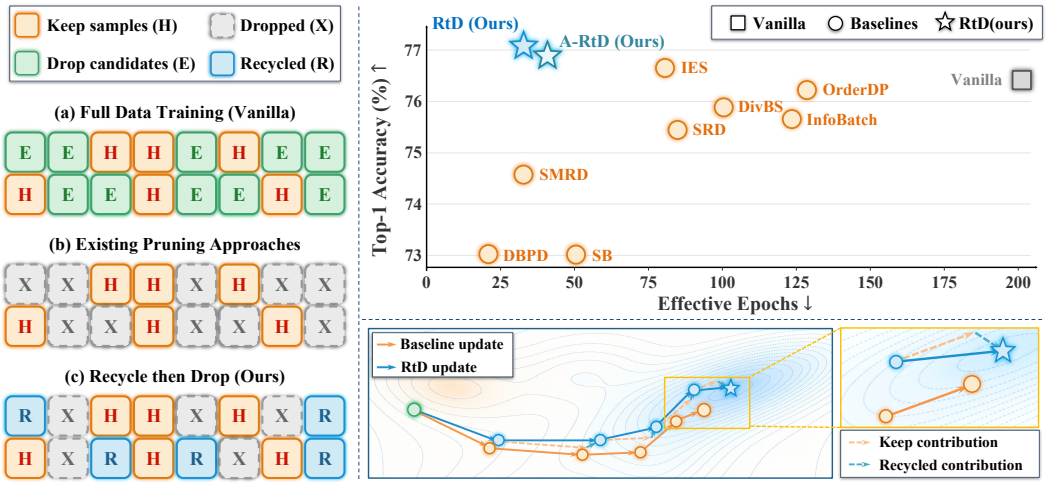


Figure 1: Selective recycling in data pruning. Left: RtD retains selected samples, recycles some exclusion candidates under stronger augmentation, and drops the remainder. Top right: top-1 accuracy versus effective epochs (EE) on CIFAR-100 with ResNet-50 (He et al., 2016). Bottom right: schematic of how training signal from recycled samples supplements the keep contribution in RtD.

Figure 1 (left) shows how RtD adds recycling to the original keep-or-drop decision. Within each minibatch, keep samples selected by the original pruning rule are trained with standard augmentation. A fraction of the drop candidates is selected uniformly, requiring no additional ranking. These candidates receive stronger augmentation than that used for scoring and are trained with keep samples in the same update. The schematic in Figure 1 (bottom right) illustrates how training signal from these recycled samples supplements the keep contribution. For the remaining candidates, backpropagation is skipped. The recycle ratio controls how many extra samples undergo backpropagation to fit the training budget. Because RtD uses the original rule's keep set and drop candidates, it can be combined with different pruning criteria while preserving their keep decisions.

RtD avoids backward passes on dropped samples, but the cost of deciding which samples to drop remains. This decision can require loading and scoring samples that will not be used for training. To reduce this repeated cost, we introduce Amortized RtD (A-RtD), which periodically updates pruning decisions and reuses them until the next update. During reuse epochs, a new recycled subset is sampled from the cached exclusion candidates before data loading. Only the keep and recycled sets are then loaded and processed for training, without recomputing pruning scores. As the model continues to learn, the update period controls the computational savings from reuse and how often selection reflects the model's changing learning state.

To assess whether recycling improves accuracy while retaining computational savings from pruning, we evaluate RtD and A-RtD across datasets, architectures, and augmentation settings. On CIFAR-100, Figure 1 (top right) shows that RtD achieves higher accuracy than full data training while using over 80% fewer samples for backpropagation. To assess time savings from decision reuse, we measure training time. On CIFAR (Krizhevsky, 2009), A-RtD reduces training time by up to 54% while achieving higher accuracy than full data training. Applying RtD to InfoBatch (Qin et al., 2023) and IES (Yuan et al., 2025) also improves accuracy while retaining their original selection criteria. Beyond classification, both methods outperform full data training on object detection and semantic segmentation with less backpropagation. Further comparisons of candidate selection and augmentation show that accuracy gains can persist even when the augmentation policy is held fixed.

Contributions. Our main contributions are summarized as follows:

- We propose RtD, which extends binary pruning by selectively recycling exclusion candidates under stronger augmentation while retaining savings from dropping the rest.
- We introduce A-RtD, an amortized variant that reuses pruning decisions to reduce repeated scoring and input processing, further shortening end-to-end training time.
- We validate accuracy and efficiency gains across multiple datasets, architectures, augmentation settings, and tasks, and demonstrate compatibility with existing pruning methods.

2 Related Work

Data pruning reduces training costs by selecting subsets (Sorscher et al., 2022; Coleman et al., 2019; Mirzasoleiman et al., 2020). Static methods select fixed subsets using random sampling, prediction error (Paul et al., 2021), forgetting events (Toneva et al., 2018), uncertainty (He et al., 2024), data coverage (Zheng et al., 2022), or representativeness (Xia et al., 2022). Dynamic methods update selection during training using loss (Jiang et al., 2019) or sample redundancy (Hong et al., 2024). InfoBatch (Qin et al., 2023) randomly prunes low-loss samples and rescales gradients, while IES (Yuan et al., 2025) uses changes in training loss to skip backpropagation. OrderDP (Jin et al., 2026) selects the highest-loss samples from a uniformly sampled pool and updates only their scores. PDD (Sathiyanarayanan et al., 2025) combines selection by confidence or random sampling with a final stage of training on all samples. These methods determine which samples are used for training in each update. RtD preserves the original keep decisions and recycles a fraction of the drop candidates under stronger augmentation to obtain training signal in the same update.

Data augmentation. Augmentation has progressed from standard transformations, including random cropping, random resized cropping (RRC), horizontal flipping, and Gaussian noise, to learned policies such as AutoAugment (Cubuk et al., 2019). RandAugment (Cubuk et al., 2020) and TrivialAugment (Müller & Hutter, 2021) simplify policy design through randomized transformations. Random Erasing (Zhong et al., 2020) masks regions, CutMix (Yun et al., 2019) mixes image patches and the corresponding labels, and AugMix (Hendrycks et al., 2020) combines augmentation chains to improve robustness. In semisupervised learning, predictions on augmented inputs are also used to construct training targets. FixMatch (Sohn et al., 2020) uses confident predictions on weakly augmented unlabeled images as pseudo labels to supervise their strongly augmented versions.

Adaptive data augmentation. Adaptive augmentation uses sample information (Gong et al., 2021) or model information (Suzuki, 2022; Yang et al., 2025a) to choose transformations. Sample difficulty also guides training order (Bengio et al., 2009; Kumar et al., 2010), selection (Shrivastava et al., 2016), and loss weighting (Lin et al., 2017). SapAugment (Hu et al., 2021) uses loss as a measure of difficulty and applies stronger augmentation to samples with lower loss. Yang et al. (2025b) select samples for augmentation to compensate for the loss of data diversity caused by selection. RtD asks whether exclusion candidates can still offer training value under stronger augmentation and recycles some of them while retaining computational savings from dropping the rest.

3 Problem Setup and Motivation

3.1 Preliminaries

Dynamic Data Pruning. Dynamic data pruning determines which samples are used for parameter updates during training (Qin et al., 2023; Sathiyanarayanan et al., 2025). Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be the training set and $\mathcal{B}_{t,b}$ the sample indices in the bth minibatch of epoch t. We denote the model parameters immediately before this minibatch by $\theta_{t,b}$. To score a sample $i \in \mathcal{B}_{t,b}$ with the current model, we apply a stochastic augmentation and use the resulting input:

$$\tilde{x}_{i,t,b} = a_{i,t,b}(x_i), \qquad s_{i,t,b} = S\big(f_{\theta_{t,b}}, \tilde{x}_{i,t,b}, y_i\big), \qquad a_{i,t,b} \sim \mathcal{A}_{\mathrm{std}}. \tag{1}$$

Here, $\mathcal{A}_{\mathrm{std}}$ is the training augmentation distribution, and S may use loss, confidence, or margin. We define S so that larger scores favor exclusion. A binary rule with threshold τ_t is

$$\pi^{\mathrm{bin}}_{i,t,b} = \begin{cases} \mathrm{keep}, & s_{i,t,b} < \tau_t, \\ \mathrm{drop}, & s_{i,t,b} \geq \tau_t, \end{cases} \tag{2}$$

where only samples assigned to *keep* are used for backpropagation in the current update.

Effective Epochs. Pruning changes the number of samples used for backpropagation, so epoch counts alone do not capture the amount of training. Let $\mathcal{U}_{t,b}$ denote the samples used for backpropagation in minibatch b of epoch t. Following Sathiyanarayanan et al. (2025), *Effective Epochs* (EE) counts these samples across training and normalizes the total by the dataset size:

$$\mathrm{EE} = \frac{1}{N} \sum_{t=1}^{T} \sum_{b} |\mathcal{U}_{t,b}|. \tag{3}$$

Using every sample once per epoch for T epochs gives $\mathrm{EE} = T$. EE does not account for the time spent on scoring and routing, so it measures total sample usage rather than wall-clock training time.

3.2 AUGMENTATION-DEPENDENT REDUNDANCY

The pruning score reflects how easily the current model handles the augmented input. Thus, $s_{i,t,b} \geq \tau_t$ means that this input meets the drop criterion, not that the original image is redundant. The same image may provide a different training signal under another augmentation. Instead of discarding all drop candidates, we select a small subset and recycle it for training under a different augmentation.

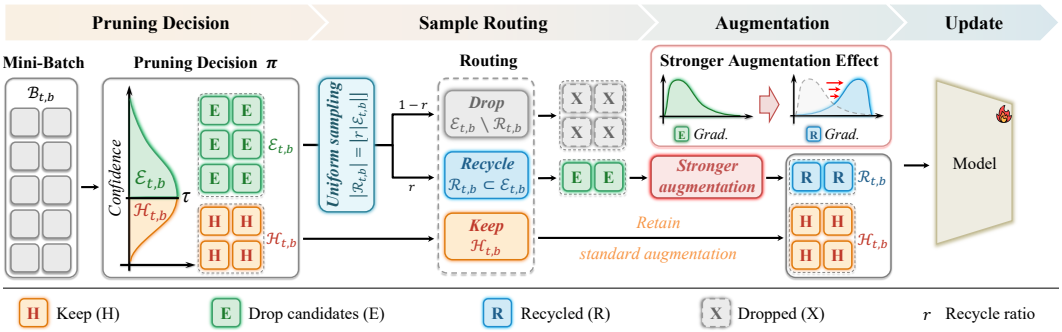


Figure 2: Recycle Then Drop (RtD). Binary pruning partitions $\mathcal{B}_{t,b}$ into keep samples $\mathcal{H}_{t,b}$ and drop candidates $\mathcal{E}_{t,b}$. RtD samples $\mathcal{R}_{t,b} \subseteq \mathcal{E}_{t,b}$ at recycle ratio r and applies stronger augmentation to this subset. Samples in $\mathcal{H}_{t,b}$ retain standard augmentation, while $\mathcal{E}_{t,b} \setminus \mathcal{R}_{t,b}$ is excluded from the update.

4 METHOD

As discussed in Section 3, a drop decision on one augmented input does not establish that the original image is redundant. To address this limitation, we propose *Recycle Then Drop* (RtD), which trains a subset of drop candidates with stronger augmentation. As illustrated in Figure 2, keep samples retain standard augmentation, recycled samples receive stronger augmentation, and the remaining candidates are excluded from each minibatch update. To reduce repeated scoring, *Amortized RtD* (A-RtD) refreshes the pruning decisions every K pruning epochs and reuses them between refreshes.

4.1 RECYCLE THEN DROP

Our default pruning score is the ground-truth-class probability $s_{i,t,b} = [\mathrm{softmax}(f_{\theta_{t,b}}(\tilde{x}_{i,t,b}))]_{y_i}$. Following the binary decision in Eq. (2), we define the keep set and drop candidates in $\mathcal{B}_{t,b}$ as

$$\mathcal{H}_{t,b} = \{i \in \mathcal{B}_{t,b} \mid \pi^{\mathrm{bin}}_{i,t,b} = \mathrm{keep}\}, \qquad \mathcal{E}_{t,b} = \mathcal{B}_{t,b} \setminus \mathcal{H}_{t,b}. \tag{4}$$

RtD retains the keep set and uniformly samples $\lfloor r|\mathcal{E}_{t,b}| \rfloor$ drop candidates without replacement to form the recycled subset $\mathcal{R}_{t,b}$, where $r \in [0,1]$ is the recycle ratio. The selected set $\mathcal{U}_{t,b} = \mathcal{H}_{t,b} \cup \mathcal{R}_{t,b}$ is used for the current update. This extends the binary decision to the routing rule

$$\pi^{\mathrm{RtD}}_{i,t,b} = \begin{cases} \mathrm{keep}, & i \in \mathcal{H}_{t,b}, \\ \mathrm{recycle}, & i \in \mathcal{R}_{t,b}, \\ \mathrm{drop}, & i \in \mathcal{E}_{t,b} \setminus \mathcal{R}_{t,b}. \end{cases} \tag{5}$$

Keep samples use the standard augmented input $\tilde{x}_{i,t,b}$ from Section 3.1. Recycled samples use $\hat{x}_{i,t,b} = a^{\mathrm{strong}}_{i,t,b}(x_i)$ with $a^{\mathrm{strong}}_{i,t,b} \sim \mathcal{A}_{\mathrm{strong}}$. For $\mathcal{U}_{t,b} \neq \emptyset$, we average their task losses ℓ:

$$\mathcal{L}^{\mathrm{RtD}}_{t,b}(\theta_{t,b}) = \frac{\sum_{i \in \mathcal{H}_{t,b}} \ell\big(f_{\theta_{t,b}}(\tilde{x}_{i,t,b}), y_i\big) + \sum_{i \in \mathcal{R}_{t,b}} \ell\big(f_{\theta_{t,b}}(\hat{x}_{i,t,b}), y_i\big)}{|\mathcal{U}_{t,b}|}. \tag{6}$$

Holding the selection fixed, the optimizer uses the loss gradient to update $\theta_{t,b}$ to $\theta_{t,b+1}$. If the selected set is empty, the update is skipped. Setting $r = 0$ recovers binary pruning, whereas $r = 1$ recycles all drop candidates. We retain any full data revision stage in the original pruning schedule.

4.2 AMORTIZED RECYCLE THEN DROP

RtD skips backpropagation for unrecycled candidates but still evaluates their pruning decisions in every pruning epoch. A-RtD reduces this remaining cost by refreshing decisions every K pruning epochs while retaining minibatch recycling. Let $\mathcal{T}_K$ denote the refresh epochs, beginning with the first pruning epoch. At each refresh, a sample's decision is evaluated using $\theta_{t,b}$ immediately before its minibatch $\mathcal{B}_{t,b}$ is processed. The cached decision is

$$m_i^{(t)} = \begin{cases} \mathbf{1}\left[\pi_{i,t,b}^{\text{bin}} = \text{drop}\right], & t \in \mathcal{T}_K, \quad i \in \mathcal{B}_{t,b}, \\ m_i^{(t-1)}, & t \notin \mathcal{T}_K. \end{cases} \tag{7}$$

Here, 0 denotes keep and 1 denotes drop. During reuse, we partition each original minibatch as

$$\mathcal{H}_{t,b} = \{i \in \mathcal{B}_{t,b} \mid m_i^{(t)} = 0\}, \qquad \mathcal{E}_{t,b} = \mathcal{B}_{t,b} \setminus \mathcal{H}_{t,b}. \tag{8}$$

We follow Section 4.1 for minibatch sampling and loss, using fresh augmentations. $K = 1$ recovers RtD. We use $K = 5$ by default. See Appendices B and D for the algorithm and overhead analysis.

4.3 GRADIENT UPDATE UNDER RECYCLING

RtD and A-RtD differ in refresh frequency but use the same loss formulation in Eq. (6). With routing and augmentations fixed, their common gradient separates into the keep and recycled terms:

$$\nabla_\theta \mathcal{L}_{t,b}^{\text{RtD}} = \frac{1}{|\mathcal{U}_{t,b}|} \left(\sum_{i \in \mathcal{H}_{t,b}} g_{i,t,b}^{\text{std}} + \sum_{i \in \mathcal{R}_{t,b}} g_{i,t,b}^{\text{strong}} \right). \tag{9}$$

Here, $g_{i,t,b}^a$ denotes the task loss gradient for sample i under augmentation a, and $\mu_{\mathcal{S},t,b}^a = |\mathcal{S}|^{-1} \sum_{i \in \mathcal{S}} g_{i,t,b}^a$ its mean over nonempty $\mathcal{S} \subseteq \mathcal{U}_{t,b}$. Then Eq. (9) becomes:

$$\nabla_\theta \mathcal{L}_{t,b}^{\text{RtD}} = \frac{|\mathcal{H}_{t,b}|}{|\mathcal{U}_{t,b}|} \mu_{\mathcal{H}_{t,b},t,b}^{\text{std}} + \frac{|\mathcal{R}_{t,b}|}{|\mathcal{U}_{t,b}|} \mu_{\mathcal{R}_{t,b},t,b}^{\text{strong}}. \tag{10}$$

Binary pruning uses only the keep mean, while recycling adds the recycled group mean. Stronger augmentation can alter both its magnitude and direction, supplementing the update to help learning and improve performance. Appendix C provides a detailed analysis of these gradient contributions.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We evaluate RtD and A-RtD on CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) using EfficientNet-B0 (Tan & Le, 2019), MobileNet-V2 (Sandler et al., 2018), ResNet-50 (He et al., 2016), TinyViT (Wu et al., 2022), and ViT-Ti (Dosovitskiy et al., 2020; Touvron et al., 2021). Additional experiments assess ImageNet-1K scalability (Russakovsky et al., 2015), compatibility with other pruning methods, and RtD's applicability to object detection and semantic segmentation. Our default CIFAR setup follows Sathiyanarayanan et al. (2025), using Resize(224) and Normalize as $\mathcal{A}_{\text{std}}$ (basic augmentation). Recycled samples additionally receive HorizontalFlip and RandAugment (Cubuk et al., 2020) ($N = 2$, $M = 9$). To also assess pruning under commonly used augmentation, Section 5.3 compares methods with resizing, random cropping, horizontal flipping, and normalization following Qin et al. (2023) (standard augmentation). Comparisons use matched settings, with other augmentation settings stated explicitly. Defaults are $\tau = 0.3$ and $r = 0.1$ for RtD and $K = 5$ for A-RtD. Within each experiment, both methods share the objective, schedule, and final full-data revision stage. We report top-1 test accuracy and effective epochs (EE) for classification, and measured wall-clock time separately. Appendix A provides augmentation and implementation details.

5.2 EVALUATION ACROSS BACKBONES

Table 1 evaluates recycling across three CNN and two Transformer backbones on CIFAR-10 and CIFAR-100 under basic augmentation. We compare RtD and A-RtD with full data training (Vanilla), DBPD, SRD, SMRD (Sathiyanarayanan et al., 2025), and InfoBatch (Qin et al., 2023). Both methods achieve higher accuracy than Vanilla in all ten settings while reducing the number of samples used for backpropagation by over 80%. This consistency supports robustness across the evaluated CNN and Transformer backbones. A-RtD stays within one percentage point of RtD and above Vanilla with refreshes every five pruning epochs, indicating that periodic reuse retains the accuracy gains. Appendix E.1 provides ResNet-50 training trajectories on CIFAR-10 and CIFAR-100.

Table 1: Evaluation across five backbones on CIFAR-10/100 under basic augmentation. Entries report top-1 accuracy (%) / EE. Best and second-best accuracies are shown in **bold** and underlined.

Method	EffNet-B0	MNet-V2	ResNet-50	TinyViT	ViT-Ti
CIFAR-10					
Vanilla	90.83 / 200.00	87.63 / 200.00	88.54 / 200.00	86.16 / 200.00	73.36 / 200.00
DBPD	86.24 / 8.61	82.89 / 7.77	84.38 / 5.64	82.14 / 5.15	62.41 / 14.39
SRD	89.43 / 84.28	87.30 / 84.28	88.53 / 84.28	85.84 / 84.28	72.64 / 84.28
SMRD	88.83 / 13.48	86.67 / 14.21	86.29 / 12.01	86.26 / 10.22	70.82 / 24.46
InfoBatch	90.46 / 117.21	86.93 / 117.88	88.20 / 116.81	85.87 / 116.90	71.08 / 122.98
RtD	92.34 / 24.23	90.54 / 24.18	**92.37** / 23.07	**91.81** / 22.59	77.23 / 31.25
A-RtD	**92.72** / 26.02	**90.78** / 26.34	92.03 / 25.12	91.44 / 24.69	**77.77** / 33.14
CIFAR-100					
Vanilla	68.22 / 200.00	61.04 / 200.00	63.46 / 200.00	56.81 / 200.00	41.42 / 200.00
DBPD	63.88 / 17.63	55.93 / 18.96	58.57 / 11.35	57.17 / 9.39	41.49 / 16.54
SRD	67.68 / 84.28	60.64 / 84.28	62.03 / 84.28	57.94 / 84.28	41.82 / 84.28
SMRD	67.12 / 27.86	62.36 / 27.84	62.12 / 18.14	58.59 / 15.56	42.37 / 23.64
InfoBatch	66.68 / 121.05	59.99 / 122.34	61.53 / 119.37	56.58 / 118.30	41.56 / 122.01
RtD	72.24 / 31.69	**67.64** / 33.33	**71.68** / 28.78	**68.91** / 26.37	48.42 / 33.99
A-RtD	**72.36** / 35.17	67.41 / 36.80	71.19 / 31.73	67.92 / 30.78	**48.66** / 37.97

5.3 Comparison with Existing Methods

Table 2 compares RtD and A-RtD with full data training and static and dynamic pruning methods on CIFAR-10 and CIFAR-100 with EfficientNet-B0 and ResNet-50. We use the standard augmentation protocol defined in Section 5.1 for 200 epochs, adding RandAugment and Random Erasing to recycled samples. On EfficientNet-B0, RtD and A-RtD achieve the two highest accuracies on both datasets while reducing the number of samples used for backpropagation by over 80% relative to Vanilla. On CIFAR-100 with ResNet-50, RtD improves accuracy over IES by 0.27 percentage points while reducing the number of samples used for backpropagation by approximately 56%. These results extend the accuracy gains and backpropagation savings to training with standard augmentation.

Table 2: Pruning comparisons on CIFAR-10/100 under standard augmentation. Entries report top-1 accuracy (%) / EE. SB denotes Selective Backpropagation.

		CIFAR-10		CIFAR-100	
	Method	EffNet-B0	ResNet-50	EffNet-B0	ResNet-50
Static	Random	92.02 / 100.0	91.78 / 100.0	68.32 / 100.0	70.21 / 100.0
	EL2N (Paul et al., 2021)	93.03 / 100.0	93.32 / 100.0	63.07 / 100.0	63.89 / 100.0
	Forgetting (Toneva et al., 2018)	93.10 / 100.0	93.65 / 100.0	65.46 / 100.0	68.36 / 100.0
	CCS (Zheng et al., 2022)	90.40 / 100.0	90.86 / 100.0	66.94 / 100.0	68.47 / 100.0
	Moderate (Xia et al., 2022)	92.10 / 100.0	92.52 / 100.0	67.76 / 100.0	70.22 / 100.0
	Dyn-Unc (He et al., 2024)	93.68 / 100.0	93.94 / 100.0	64.50 / 100.0	67.96 / 100.0
Dynamic	SB (Jiang et al., 2019)	93.43 / 50.7	93.93 / 49.6	75.23 / 50.8	73.01 / 50.8
	DBPD (Sathiyanarayanan et al., 2025)	90.26 / 11.9	90.60 / 9.6	74.34 / 24.7	73.00 / 21.8
	SRD (Sathiyanarayanan et al., 2025)	94.16 / 84.3	94.45 / 84.3	74.98 / 84.3	75.52 / 84.3
	SMRD (Sathiyanarayanan et al., 2025)	91.62 / 18.1	91.16 / 17.9	74.72 / 33.1	74.61 / 32.1
	InfoBatch (Qin et al., 2023)	94.34 / 118.9	94.49 / 118.7	74.51 / 124.1	75.81 / 124.1
	DivBS (Hong et al., 2024)	94.29 / 100.0	94.43 / 100.0	76.05 / 100.0	75.94 / 100.0
	IES (Yuan et al., 2025)	94.22 / 51.4	94.58 / 47.3	75.70 / 82.3	76.79 / 79.8
	OrderDP (Jin et al., 2026)	94.14 / 127.5	94.16 / 127.5	75.40 / 127.5	76.14 / 127.5
	RtD (ours)	**94.64** / 26.4	**94.99** / 25.3	76.63 / 36.9	**77.06** / 35.0
	A-RtD (ours)	94.58 / 28.2	94.81 / 27.6	**76.74** / 39.3	76.92 / 38.0
	Vanilla	94.37 / 200.0	94.46 / 200.0	76.22 / 200.0	76.39 / 200.0

5.4 PRACTICAL EFFICIENCY

Table 3 compares training time under basic augmentation. Skipping backpropagation still leaves the cost of scoring samples. RtD reduces training time by 25% on both datasets but still scores every sample in each pruning epoch. A-RtD skips this scoring during reuse epochs, saving 54% on CIFAR-10 and 52% on CIFAR-100 relative to Vanilla while achieving the highest accuracy among all compared methods. A-RtD trains faster than RtD despite using slightly more samples for backpropagation. This shows why reducing the cost of sample scoring matters for actual training time. Appendix D separately breaks down these costs.

Table 3: CIFAR-10/100 runtime with EfficientNet-B0. Entries report top-1 accuracy (%) / EE and wall-clock time in minutes. ↓ denotes the percentage reduction in time relative to Vanilla.

Dataset	CIFAR-10		CIFAR-100	
Method	Acc. / EE	Wall-clock (↓)	Acc. / EE	Wall-clock (↓)
Vanilla	90.83 / 200.0	269.5 (–)	68.22 / 200.0	269.7 (–)
DBPD	86.24 / 8.6	156.1 (42%)	63.88 / 17.6	165.9 (38%)
SRD	89.43 / 84.3	171.2 (36%)	67.68 / 84.3	166.5 (38%)
SMRD	88.83 / 13.5	190.9 (29%)	67.12 / 27.9	187.3 (31%)
InfoBatch	90.46 / 117.2	<u>141.0</u> (<u>48%</u>)	66.68 / 121.1	<u>145.3</u> (<u>46%</u>)
RtD	<u>92.34</u> / 24.2	203.4 (25%)	<u>72.24</u> / 31.7	201.1 (25%)
A-RtD	**92.72** / 26.0	**125.0** (**54%**)	**72.36** / 35.2	**128.6** (**52%**)

5.5 SCALING TO IMAGENET-1K

Table 4 evaluates whether recycling remains effective when training scales to ImageNet-1K. We train ResNet-50 (He et al., 2016) for 100 epochs under standard ImageNet augmentation and measure wall-clock time on an RTX 4090. Appendix A provides the full protocol. RtD and A-RtD both outperform Vanilla while reducing the number of samples used for backpropagation by 40.5% and 37.9%, respectively. A-RtD also reduces training time by 32.9% relative to Vanilla and by 19.3% relative to RtD. Among the methods that match or outperform Vanilla, A-RtD is the fastest. SRD is faster than A-RtD, but its accuracy falls below Vanilla, showing the trade-off between speed and accuracy. Periodic decision reuse maintains higher accuracy than Vanilla while reducing training wall-clock time relative to RtD. The accuracy and efficiency pattern observed on CIFAR also appears in this larger-scale training setting.

Table 4: ImageNet-1K results with ResNet-50 over 100 epochs.

Method	Acc. / EE	Wall (h)
Vanilla	75.17 / 100.0	30.98
DBPD	72.62 / **51.5**	24.83
SRD	74.60 / 63.8	**19.58**
SMRD	73.89 / <u>52.7</u>	25.18
InfoBatch	75.14 / 70.8	21.31
RtD	**75.42** / 59.5	25.78
A-RtD	<u>75.39</u> / 62.1	<u>20.80</u>

5.6 COMPATIBILITY WITH EXISTING PRUNING METHODS

To test whether recycling benefits different pruning methods, we add it to InfoBatch (Qin et al., 2023), IES (Yuan et al., 2025), and DBPD (Sathiyanarayanan et al., 2025) without changing their exclusion rules. For each method, we first recycle a fraction of the excluded samples under standard augmentation, then replace their training views with stronger augmentation. In Table 5, recycling alone improves all three methods, showing that excluded samples retain useful training signal. Stronger views improve accuracy further in every case, adding value beyond simply reusing those samples. Together, these gains show that recycling remains useful across different exclusion rules. The gains come with additional backpropagation because recycled samples return to training. Appendix E.3 reports the corresponding comparison under basic augmentation protocol.

Table 5: EfficientNet-B0 on CIFAR-100 with standard augmentation. Accuracy (%) / EE. Vanilla attains 76.22 / 200.

Recycling	Strong aug.	InfoBatch	IES	DBPD
×	×	74.51 / 124.1	75.70 / 82.3	74.34 / 24.7
✓	×	75.26 / 131.4	76.09 / 94.9	76.19 / 35.4
✓	✓	**75.87** / 131.8	**76.61** / 102.9	**76.63** / 36.9

5.7 BEYOND CLASSIFICATION

To test whether recycling applies beyond image classification, we apply RtD to object detection and semantic segmentation. Both tasks produce many predictions per image, so we define an image-level confidence from each output and keep the routing rule unchanged. Because this averaged confidence is high for most images, we use τ=0.8 instead of 0.3, with r=0.1 and K=5 (A-RtD). All methods follow the IES protocol (Yuan et al., 2025) within each task, and we report Saved BP, the percentage of samples excluded from backpropagation relative to Vanilla. Appendix A provides the details.

Object detection. We train Faster R-CNN (Ren et al., 2015) with VGG-16 (Simonyan & Zisserman, 2014) on PASCAL VOC 2007 (Everingham et al., 2007) for 30 epochs. Detection confidence is computed from class-matched predictions. In Table 6, InfoBatch and IES save about 20% of backpropagation with slightly higher mAP than Vanilla. RtD saves about twice as much, 38.7%, and gains more mAP. A-RtD saves slightly less than RtD but achieves the highest mAP and the shortest training time.

Table 6: Object detection results. Results are averaged over three seeds.

Method	mAP / BP	Wall (min)
Vanilla	67.64 ± 0.2 / –	98.4
InfoBatch	67.93 ± 0.2 / 19.7	87.7
IES	67.81 ± 0.1 / 19.5	88.3
RtD	$\underline{68.48} \pm 0.2$ / **38.7**	$\underline{87.4}$
A-RtD	**68.52** ± 0.1 / $\underline{36.7}$	**62.6**

Semantic segmentation. We train DeepLabV3 (Chen et al., 2017) with ResNet-50 (He et al., 2016) on PASCAL VOC 2012 (Everingham et al., 2015) for 50 epochs. Segmentation confidence averages the ground-truth-class probability over valid pixels. In Table 7, InfoBatch and IES save about 12% of backpropagation but do not exceed Vanilla in mean mIoU. RtD and A-RtD save about twice as much and stay above Vanilla, with RtD the most accurate and A-RtD the fastest. Across both tasks, recycling improves accuracy and reduces backpropagation beyond classification using task-specific confidence scores and the same routing rule.

Table 7: Semantic segmentation results.

Method	mIoU / BP	Wall (min)
Vanilla	75.28 ± 0.1 / –	259.7
InfoBatch	75.11 ± 0.2 / 11.5	241.3
IES	75.14 ± 0.2 / 11.6	239.1
RtD	**75.65** ± 0.2 / **24.4**	$\underline{238.7}$
A-RtD	$\underline{75.53} \pm 0.2$ / $\underline{21.9}$	**209.8**

5.8 Gradient and Routing Diagnostics

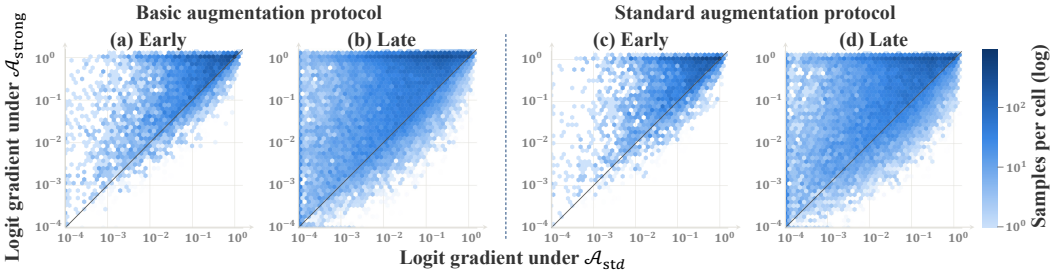


Figure 3: Recycled samples' logit gradient norms under standard and stronger augmentation during 30-epoch training. Early and Late pool epochs 2–10 and 21–29. The diagonal denotes equal norms.

Figure 3 compares logit gradient norms $g = \|\text{softmax}(z) - e_y\|_2$ of recycled candidates on CIFAR-100 with EfficientNet-B0, where z denotes logits and e_y the one-hot label vector. Although confidence under $\mathcal{A}_{\text{std}}$ marks these samples for exclusion, $\mathcal{A}_{\text{strong}}$ produces larger gradients in both protocols. Recycled samples have larger median parameter gradient norms than keep samples in Figure 4(a). Thus, exclusion under one augmentation may overlook training signal under another.

Table 8 tests whether new augmented inputs improve recycling beyond scaling the scoring input's gradient. None omits recycling, while Fresh draw generates a new augmented input. Norm matching rescales the scoring input's loss to approach the norm g under $\mathcal{A}_{\text{strong}}$ without changing direction. In both protocols, fresh standard inputs outperform norm matching, and stronger augmentation improves accuracy further. Under Standard, these gains persist despite 98.3% average norm realization and similar EE. This supports new augmented inputs as a source of useful training signal. See Appendix E.1 for details.

Table 8: EfficientNet-B0 on CIFAR-100, 200 epochs. Basic and Standard are augmentation protocols. Single seed test accuracy (%) / EE.

Recycled input	Basic	Standard
None	63.88 / 17.6	74.34 / 24.7
Norm matching	66.10 / 30.8	75.20 / 37.0
Fresh draw from $\mathcal{A}_{\text{std}}$	66.62 / 30.2	76.19 / 35.4
Fresh draw from $\mathcal{A}_{\text{strong}}$	**72.14** / 31.6	**76.63** / 36.9

To examine the direction of this added signal, we compare mean gradients of recycled and keep samples. Appendix Table 12 reports positive cosine in every measured condition. The accuracy gains and alignment provide evidence that recycling supplements the keep contribution with useful training signal that pruning omits. Finally, Figure 4(b) shows that fewer than 2% of decisions change between consecutive epochs after the early stage, supporting decision reuse in A-RtD.

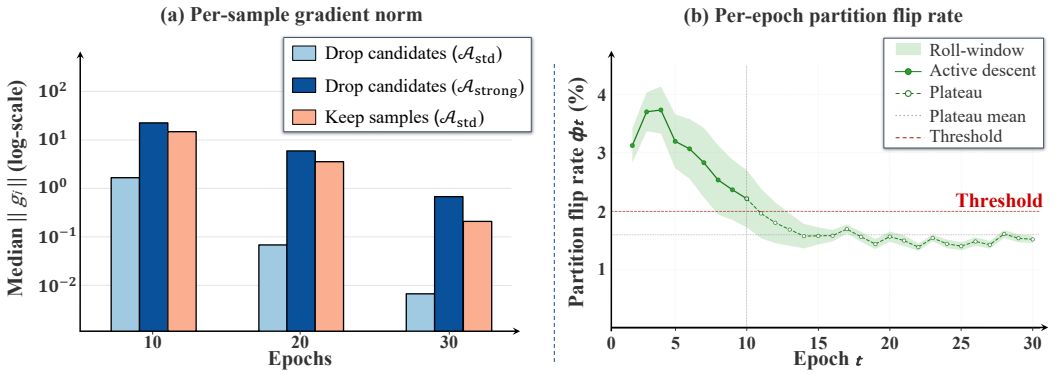


Figure 4: CIFAR-100 diagnostics with EfficientNet-B0. **(a)** Median sample gradient norms at epochs 10, 20, and 30. **(b)** Fraction of the routing decisions that change across consecutive epochs.

5.9 Ablations

RtD makes two separate decisions. The confidence score chooses which samples to revisit, and the stronger augmentation determines what those samples learn from. We isolate each decision with the other held fixed and report further placement, policy, and hyperparameter analyses in Appendix E.

Which samples should be recycled? Table 9 compares recycling from the full training set with recycling from drop candidates, using the same augmentation and the same number of recycled samples. Uniform sampling from the full training set already improves accuracy over no recycling, while sampling from drop candidates improves accuracy further on both datasets. These results show that recycling from the candidate pool defined by confidence is more effective than recycling the same number of uniformly sampled examples from the full training set.

Table 9: Recycled pool with EfficientNet-B0, r=0.1, and 30 epochs. All variants use the same augmentation and the same number of recycled samples. Values are mean ± std over three seeds.

Recycled pool	CIFAR-10	CIFAR-100
No recycling	87.41 ± 0.16	65.91 ± 0.13
Full training set	89.68 ± 0.24	67.65 ± 0.22
Drop candidates (RtD)	**90.26** ± 0.19	**68.56** ± 0.21

What contributes beyond recycling alone? Table 10 compares augmentations applied to recycled samples. Recycling under $\mathcal{A}_{\text{std}}$ already improves accuracy over no recycling. We use Gaussian noise with $\sigma = 0.49$ to match the scale of input changes from strong augmentation. This value is the empirical standard deviation of differences between standard and default strong inputs across training images. Gaussian noise adds little, whereas RandAugment and horizontal flipping yield larger improvements. Thus, recycling alone improves accuracy, while additional gains depend on the transformations applied to recycled inputs. Appendix Table 20 compares further augmentation policies.

Table 10: Augmentation applied to recycled samples with EfficientNet-B0, r=0.1, and 200 epochs. Gaussian noise uses σ=0.49.

Recycled augmentation	CIFAR-10	CIFAR-100
No recycling	86.24	63.88
$\mathcal{A}_{\text{std}}$ only	89.51	66.72
Gaussian noise	89.81	66.52
RandAugment	91.40	69.39
RandAugment + Flip	**92.34**	**72.24**

6 Conclusion

We proposed Recycle Then Drop (RtD) to recover training signal from samples excluded under one augmentation. RtD recycles a fraction of drop candidates with stronger augmentation while preserving pruning's computational savings. A-RtD reuses decisions across epochs to reduce repeated scoring. Experiments across models and visual recognition tasks show accuracy and efficiency gains. Recycling also improves the accuracy of existing pruning methods. Accuracy comparisons and gradient analyses support supplementing the keep update with useful signal from new augmented inputs. Efficient training can thus benefit from recovering signal from drop candidates.

Limitations and future work. Our method uses predefined augmentation and recycling settings. Future work will explore adapting augmentation, the recycle ratio, and the decision refresh period during training. Building on our visual recognition results, we will also extend recycling to self-supervised learning, generative modeling, and language tasks and evaluate its effectiveness.

AI USE STATEMENT

We used AI in a limited manner for grammar and style polishing only. It did not contribute to research ideation, experimental design, and analysis. The authors take full responsibility for all content.

REPRODUCIBILITY STATEMENT

The datasets, augmentation policies, backbone architectures, optimization settings, and evaluation protocols are described in Section 5 and Appendix A. The complete RtD and A-RtD training procedure is provided in Appendix B, together with additional robustness, overhead, and ablation results.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

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. *arXiv preprint arXiv:1906.11829*, 2019.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 113–123, 2019.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Keepaugment: A simple information-preserving data augmentation approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1055–1064, 2021.

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*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Muyang He, Shuo Yang, Tiejun Huang, and Bo Zhao. Large-scale dataset pruning with dynamic uncertainty. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 7713–7722. IEEE, 2024.

Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Feng Hong, Yueming Lyu, Jiangchao Yao, Ya Zhang, Ivor W Tsang, and Yanfeng Wang. Diversified batch selection for training acceleration. *arXiv preprint arXiv:2406.04872*, 2024.

Ting-Yao Hu, Ashish Shrivastava, Jen-Hao Rick Chang, Hema Swetha Koppula, Stefan Braun, Kyuyeon Hwang, Ozlem Kalinli, and Oncel Tuzel. Sapaugment: Learning a sample adaptive policy for data augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4040–4044. IEEE, 2021.

Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

Chenhan Jin, Shengze Xu, Qingsong Wang, Fan Jia, Dingshuo Chen, and Tieyong Zeng. Orderdp: A theoretically guaranteed lossless dynamic data pruning framework. *arXiv preprint arXiv:2606.08574*, 2026.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL `https://api.semanticscholar.org/CorpusID:18268744`.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022.

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

Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 774–782, 2021.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607, 2021.

Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, Zhaopan Xu, Daquan Zhou, Lei Shang, Baigui Sun, Xuansong Xie, et al. Infobatch: Lossless training speed up by unbiased dynamic data pruning. *arXiv preprint arXiv:2303.04947*, 2023.

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.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Shriram M Sathiyanarayanan, Xinyue Hao, Shihao Hou, Yang Lu, Laura Sevilla-Lara, Anurag Arnab, and Shreyank N Gowda. Progressive data dropout: An embarrassingly simple approach to faster training. *arXiv preprint arXiv:2505.22342*, 2025.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 761–769, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

Teppei Suzuki. Teachaugment: Data augmentation optimization using teacher knowledge. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10904–10914, 2022.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

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

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European conference on computer vision*, pp. 68–85. Springer, 2022.

Xiaobo Xia, Jiale Liu, Jun Yu, Xu Shen, Bo Han, and Tongliang Liu. Moderate coreset: A universal method of data selection for real-world data-efficient deep learning. In *The Eleventh International Conference on Learning Representations*, 2022.

Suorong Yang, Peijia Li, Xin Xiong, Furao Shen, and Jian Zhao. Adaaugment: A tuning-free and adaptive approach to enhance data augmentation. *IEEE Transactions on Image Processing*, 2025a.

Suorong Yang, Peng Ye, Furao Shen, and Dongzhan Zhou. When dynamic data selection meets data augmentation: Achieving enhanced training acceleration. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=XStA4AX4dK.

Suqin Yuan, Runqi Lin, Lei Feng, Bo Han, and Tongliang Liu. Instance-dependent early stopping. *arXiv preprint arXiv:2502.07547*, 2025.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. Coverage-centric coreset selection for high pruning rates. *arXiv preprint arXiv:2210.15809*, 2022.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 13001–13008, 2020.

A Implementation Details

CIFAR experiments. Unless otherwise stated, all models are trained from scratch. For CIFAR-10 and CIFAR-100, the default optimizer is AdamW with an initial learning rate of 3×10^{-4}, weight decay 0.01, batch size 32, and StepLR with γ=0.98. Baseline methods follow their official implementations and recommended settings. Unless otherwise stated, CIFAR experiments use FP32 training on a single NVIDIA RTX 3090 GPU with 24 GB memory under both CIFAR protocols.

ImageNet-1K experiments. We train ResNet-50 for 100 epochs using SGD with an initial learning rate of 0.1, cosine annealing, batch size 256, and label smoothing of 0.1. ImageNet-1K experiments in Table 4 use mixed precision on a single NVIDIA RTX 4090 GPU with 24 GB memory.

Detection and segmentation experiments. Following Yuan et al. (2025), we train Faster R-CNN (VGG-16) on VOC 2007 for 30 epochs and DeepLabV3 (ResNet-50) on VOC 2012 trainaug for 50 epochs. Detection confidence scores annotated objects using class-matched predictions. Segmentation confidence averages ground-truth-class probabilities over valid pixels. Recycling adds color jitter and Gaussian blur. We use τ=0.8, r=0.1, and K=5 for A-RtD. We report best mAP/mIoU and Saved BP, the percentage reduction in samples used for backpropagation relative to Vanilla.

Optimizer comparison. For the optimizer comparison in Appendix E, we follow InfoBatch (Qin et al., 2023) and train all models for 200 epochs with the OneCycle scheduler, using cosine annealing and 30% warm-up. All three optimizers use weight decay 5×10^{-4} and label smoothing of 0.1. SGD uses momentum 0.9, a maximum learning rate of 0.03, and batch size 128. LARS uses a maximum learning rate of 5.2 and batch size 256, while LAMB uses a maximum learning rate of 0.01 and batch size 128. For LARS and LAMB, we use the official InfoBatch implementation. Within each optimizer setting, all compared methods share the maximum learning rate and OneCycle schedule.

Fine-grained experiments. For the fine-grained classification experiments in Table 25, we initialize EfficientNet-B0 with ImageNet pretrained weights. The standard augmentation and the stronger augmentation applied to recycled samples are specified for each experiment in Table 11 below.

Table 11: Augmentations. + adds to $\mathcal{A}_{\text{std}}$; = means identical to $\mathcal{A}_{\text{std}}$. RRC/RC denote RandomResizedCrop/RandomCrop. Flip is horizontal. VOC views share the same geometry with boxes/masks.

Standard $\mathcal{A}_{\text{std}}$	**Recycled $\mathcal{A}_{\text{strong}}$**	**Experiments**
Basic CIFAR. Resize(224), Normalize.	+ Flip, RandAugment	Tables 1, 3, 8, 22, 9, 12, 18, 19, 21, 16, 17; Table 27 (ViT-B) Figures 5, 4, 6; Figure 3(a,b)
	+ Flip, (RandAugment / TrivialAugment / RandomErasing)	Table 20
	= / + Gaussian noise / + RandAugment / + Flip, RandAugment	Table 10
Standard CIFAR. Resize(224), Flip, RC(224), Normalize.	+ RandAugment, RandomErasing	Tables 2, 5, 8, 24, 14, 23, 16, 17; Table 27 (TinyViT); Figure 3(c,d)
ImageNet. RRC(224), Flip, Normalize.	+ RandAugment, RandomErasing	Table 4
Fine-grained / long-tailed. RRC(224), Flip, Normalize.	+ RandAugment, RandomErasing	Tables 25, 26
Augmentation control. RandAugment / AugMix / TrivialAugment	=	Table 15
Basic CIFAR. Resize(224), Normalize	+ RandAugment	Table 13
Detection. Flip, Resize, Normalize	+ ColorJitter, GaussianBlur	Table 6
Segmentation. RandomResize, Pad, RC(480), Flip, Normalize	+ ColorJitter, GaussianBlur	Table 7

B ALGORITHM

Algorithm 1 provides a unified training procedure for RtD and A-RtD. RtD steps are shown in black, while the A-RtD-specific cache and reuse steps are shown in blue.

Algorithm 1 Unified training procedure for RtD and A-RtD. Setting $K{=}1$ recovers RtD.

Require: data $\mathcal{D}$, initial parameters θ, configured optimizer OptStep, original pruning schedule, recycle ratio r, update period K, augmentations $\mathcal{A}_{\text{std}}$ and $\mathcal{A}_{\text{strong}}$

$m_i^{(0)} \leftarrow$ INVALID for all i
for epoch t in the original pruning schedule **do**
  if t is a pruning epoch **then**
    for original index minibatch $\mathcal{B}_{t,b}$ of $\mathcal{D}$ **do**
      $\theta_{t,b} \leftarrow \theta$
      if $t \in \mathcal{T}_K$ **then**
        $(\tilde{x}_{i,t,b}, s_{i,t,b}, \pi^{\text{bin}}_{i,t,b}) \leftarrow$ Eqs. (1)–(2), $i \in \mathcal{B}_{t,b}$
        $m_i^{(t)} \leftarrow \mathbf{1}[\pi^{\text{bin}}_{i,t,b} = \text{drop}]$, $i \in \mathcal{B}_{t,b}$
      else
        $m_i^{(t)} \leftarrow m_i^{(t-1)}$, $i \in \mathcal{B}_{t,b}$
      end if
      $\mathcal{H}_{t,b} \leftarrow \{i \in \mathcal{B}_{t,b} \mid m_i^{(t)} = 0\}$, $\mathcal{E}_{t,b} \leftarrow \mathcal{B}_{t,b} \setminus \mathcal{H}_{t,b}$
      $\mathcal{R}_{t,b} \sim \text{Unif}\{R \subseteq \mathcal{E}_{t,b} \mid |R| = \lfloor r|\mathcal{E}_{t,b}| \rfloor\}$
      $\mathcal{U}_{t,b} \leftarrow \mathcal{H}_{t,b} \cup \mathcal{R}_{t,b}$
      if $\mathcal{U}_{t,b} \neq \emptyset$ **then**
        if $t \notin \mathcal{T}_K$ **then**
          $\tilde{x}_{i,t,b} \leftarrow a_{i,t,b}(x_i)$, $a_{i,t,b} \sim \mathcal{A}_{\text{std}}$, $i \in \mathcal{H}_{t,b}$
        end if
        $\hat{x}_{i,t,b} \leftarrow a^{\text{strong}}_{i,t,b}(x_i)$, $a^{\text{strong}}_{i,t,b} \sim \mathcal{A}_{\text{strong}}$, $i \in \mathcal{R}_{t,b}$
        $\theta \leftarrow \text{OptStep}\Big(\theta_{t,b}, \nabla_{\theta_{t,b}} \mathcal{L}^{\text{RtD}}_{t,b}(\theta_{t,b})\Big)$ using Eq. (6)
      end if
    end for
  else
    Update θ under the original schedule stage
  end if
end for

C GRADIENT AND ROUTING DIAGNOSTICS

Section 5.8 shows performance gains from selective recycling with stronger augmentation, but not how recycling changes the minibatch gradient. Recycling changes both the gradient sum and the number of selected samples used to average it, so we derive the keep and recycled terms with their weights. Gradient norms omit direction, so we compare mean parameter gradients from keep and drop candidates. Figure 4(b) tracks decision changes across consecutive epochs, but A-RtD reuses decisions for K epochs. We therefore compare cached and fresh pruning decisions by cache age.

C.1 GRADIENT CONTRIBUTIONS UNDER RECYCLING

Equation (6) averages the selected losses over $\mathcal{U}_{t,b} = \mathcal{H}_{t,b} \cup \mathcal{R}_{t,b}$. Define the group counts by

$$h_{t,b} = |\mathcal{H}_{t,b}|, \qquad m_{t,b} = |\mathcal{R}_{t,b}| = \lfloor r|\mathcal{E}_{t,b}| \rfloor, \qquad n_{t,b} = |\mathcal{U}_{t,b}| = h_{t,b} + m_{t,b}. \tag{11}$$

Let $\ell_i(\theta; \mathcal{U}_{t,b})$ be sample i's loss in the joint forward pass over $\mathcal{U}_{t,b}$, with parameter gradient

$$g^{\mathcal{U}}_{i,t,b} = \nabla_\theta \ell_i(\theta; \mathcal{U}_{t,b})|_{\theta=\theta_{t,b}}. \tag{12}$$

When $n_{t,b} = 0$, the update is skipped. For $n_{t,b} > 0$, differentiating Eq. (6) with the selected set fixed gives

$$\begin{aligned} g_{t,b}^{\mathrm{RtD}} &= \nabla_\theta \left(\frac{1}{n_{t,b}} \sum_{i \in \mathcal{U}_{t,b}} \ell_i(\theta; \mathcal{U}_{t,b}) \right) \Bigg|_{\theta=\theta_{t,b}} \\ &= \frac{1}{n_{t,b}} \sum_{i \in \mathcal{U}_{t,b}} g_{i,t,b}^{\mathcal{U}} \\ &= \frac{1}{n_{t,b}} \left(\sum_{i \in \mathcal{H}_{t,b}} g_{i,t,b}^{\mathcal{U}} + \sum_{i \in \mathcal{R}_{t,b}} g_{i,t,b}^{\mathcal{U}} \right). \end{aligned} \tag{13}$$

For each nonempty group, define its mean gradient by

$$\mu_{S,t,b}^{\mathcal{U}} = \frac{1}{|S|} \sum_{i \in S} g_{i,t,b}^{\mathcal{U}}, \qquad S \in \{\mathcal{H}_{t,b}, \mathcal{R}_{t,b}\}.$$

When both groups are nonempty, substituting these means into Eq. (13) gives

$$\begin{aligned} g_{t,b}^{\mathrm{RtD}} &= \frac{1}{n_{t,b}} \left(h_{t,b} \mu_{\mathcal{H}_{t,b},t,b}^{\mathcal{U}} + m_{t,b} \mu_{\mathcal{R}_{t,b},t,b}^{\mathcal{U}} \right) \\ &= \frac{h_{t,b}}{n_{t,b}} \mu_{\mathcal{H}_{t,b},t,b}^{\mathcal{U}} + \frac{m_{t,b}}{n_{t,b}} \mu_{\mathcal{R}_{t,b},t,b}^{\mathcal{U}}. \end{aligned} \tag{14}$$

Equation (14) weights the keep and recycled means by their shares of the selected minibatch. Recycling thus supplements the update with gradients that binary pruning would omit. We next assess the direction of this added signal relative to the keep mean, which the weights alone do not determine.

C.2 Aggregate Gradient Alignment

Figure 4(a) reports larger median parameter gradient norms for drop candidates under stronger augmentation than for keep samples. RtD adds gradients from these candidates to the keep term, so their magnitude alone does not show whether the added signal also points along the keep direction. We compare the two group means at fixed checkpoints. At checkpoint t, let $g_{i,t}^{a}$ be the measured parameter gradient for sample i under augmentation $a \in \{\mathrm{std}, \mathrm{strong}\}$. For any nonempty subset S, its mean parameter gradient is

$$\mu_{S,t}^{a} = \frac{1}{|S|} \sum_{i \in S} g_{i,t}^{a}. \tag{15}$$

For a fixed analysis subset S, let $\mathcal{H}_{S,t}$ and $\mathcal{E}_{S,t}$ denote its keep and drop candidate groups. We measure the alignment of their nonzero means by cosine similarity

$$\alpha_{S,t} = \frac{\left\langle \mu_{\mathcal{H}_{S,t},t}^{\mathrm{std}}, \mu_{\mathcal{E}_{S,t},t}^{\mathrm{strong}} \right\rangle}{\left\| \mu_{\mathcal{H}_{S,t},t}^{\mathrm{std}} \right\|_2 \left\| \mu_{\mathcal{E}_{S,t},t}^{\mathrm{strong}} \right\|_2}. \tag{16}$$

Under the basic CIFAR protocol, we evaluate 10,000 fixed samples and average three RandAugment applications per checkpoint. All 24 mean cosines are positive and range from 0.16 to 0.61. Because RtD selects recycled samples from this candidate pool, the result shows that the available signal has a component along the keep mean at the evaluated checkpoints. The controlled comparison in Table 8 shows that selective recycling with stronger augmentation outperforms the tested controls in both protocols. Together, this alignment and the controlled performance gains support recycling as a way to supplement the keep update with useful training signal that binary pruning would omit.

Table 12: Cosine similarity of mean gradients for keep samples under standard augmentation and drop candidates under stronger augmentation with EfficientNet-B0.

Dataset	Epoch	τ=0.3	τ=0.5	τ=0.7	τ=0.9
CIFAR-100	10	+0.33	+0.27	+0.30	+0.16
	20	+0.17	+0.18	+0.23	+0.17
	30	+0.32	+0.34	+0.48	+0.53
CIFAR-10	10	+0.54	+0.61	+0.59	+0.57
	20	+0.33	+0.37	+0.23	+0.27
	30	+0.18	+0.27	+0.39	+0.45

C.3 Cached Pruning Decision Diagnostics for A-RtD

A-RtD refreshes pruning decisions every K epochs and reuses them between refreshes. If a cached decision differs from one made by the current model, the same sample would receive a different keep or drop candidate assignment. We measure how this disagreement changes over the reuse interval. Let t_{ref} be the most recent refresh epoch. The number of epochs since refresh is

$$a_t = t - t_{\mathrm{ref}} \in \{0, \dots, K-1\}. \tag{17}$$

The decision is refreshed at $a_t = 0$ and reused at $a_t = 1, \dots, 4$ when $K = 5$. For sample i in minibatch $\mathcal{B}_{t,b}$, let $m^{\mathrm{C}}_{i,t}$ be the cached drop decision used for routing. We also compute a fresh decision $\widehat{m}^{\mathrm{F}}_{i,t,b}$ with the current model just before the update, solely for comparison. The fraction of the N training samples with different decisions in epoch t is

$$\delta_t = \frac{1}{N} \sum_b \sum_{i \in \mathcal{B}_{t,b}} \mathbf{1}\left[m^{\mathrm{C}}_{i,t} \neq \widehat{m}^{\mathrm{F}}_{i,t,b}\right]. \tag{18}$$

The extra evaluation does not change routing or training. It checks whether the cached drop decisions agree with decisions made by the current model. The final full data revision uses no pruning decisions, so we split the preceding 199 epochs into three nearly equal ranges. Each cell reports the mean $\pm$ standard deviation over three seeds after averaging epochs within each seed. Table 13 shows that disagreement falls as training progresses, although it rises with time since refresh early on. By the final third, it stays near 2% across all four reuse epochs. The roughly 98% late agreement supports reusing cached decisions instead of recomputing them. Consistent with this, A-RtD stays within one percentage point of RtD.

Table 13: Disagreement (%) between cached and fresh drop decisions on CIFAR-100 with EfficientNet-B0.

Epochs	$a_t = 1$	$a_t = 2$	$a_t = 3$	$a_t = 4$
1–67	17.41 ± 0.27	20.93 ± 0.35	23.01 ± 0.31	24.55 ± 0.27
68–133	4.23 ± 0.22	4.69 ± 0.23	4.96 ± 0.22	5.07 ± 0.21
134–199	1.91 ± 0.14	1.92 ± 0.16	1.99 ± 0.14	2.04 ± 0.14

D Overhead Analysis

A lower EE does not ensure a proportional reduction in wall clock time because pruning still evaluates confidence. Under the standard augmentation protocol, RtD shortens training updates but still evaluates confidence every pruning epoch. With $K = 5$, A-RtD uses a similar update fraction while skipping this evaluation during reuse epochs. Table 14 shows an 80% reduction in average confidence evaluation time. Across the full schedule, RtD cuts wall clock time by about 25% and A-RtD by about 45% relative to Vanilla. Decision reuse thus contributes to the additional time saving.

Table 14: CIFAR-100 runtime for EfficientNet-B0 over 200 epochs on one RTX 4090. q_t is the update fraction. Direct totals include residual overhead.

Method	Epoch type	q_t	Confidence (s)	Input + aug. (s)	Training update (s)	Total (s)
Vanilla	Standard	1.000	0.00	0.50	62.66	66.28
RtD	Pruning	0.181	17.24	6.30	21.44	49.47
	Revision	1.000	0.00	0.44	62.98	66.53
A-RtD	Update	0.192	17.22	6.07	21.78	49.52
	Reuse	0.193	0.00	6.46	22.24	33.02
	Revision	1.000	0.00	0.53	62.74	66.36
	Per-cycle avg.	0.192	**3.44**	6.38	22.15	**36.32**

Method	Total wall (200 ep, s)	Ratio to Vanilla
Vanilla	13255.8	1.00×
RtD	9910.5	0.75×
A-RtD (K=5)	7296.8	0.55×

E Additional Experiments

We present controls for selective recycling, examine its design and sensitivity, test compatibility with other pruning methods and robustness across optimizers and seeds, and cover additional datasets and backbones. Unless stated otherwise, RtD uses $r = 0.1$ and $\tau = 0.3$, and A-RtD uses $K = 5$.

E.1 Controls for Selective Recycling

Identical augmentation. To assess whether differences in augmentation explain RtD's accuracy gain, Table 15 compares RtD with full data training using AugMix (Hendrycks et al., 2020), TrivialAugment (Müller & Hutter, 2021), and RandAugment (Cubuk et al., 2020). For the RandAugment comparison, Vanilla and both keep and recycled samples in RtD use the identical configuration $(N = 2, M = 9)$. RtD achieves the highest accuracy among the evaluated settings with about 81% fewer backpropagated samples than Vanilla. The accuracy advantage under identical augmentation supports a contribution from RtD's routing beyond the augmentation policy.

Table 15: Augmentation comparison on CIFAR-100 with EfficientNet-B0 over 200 epochs. Accuracy is in percent.

Method	Acc. (%)	EE
Vanilla + AugMix	73.01	200.0
Vanilla + TrivialAugment	74.00	200.0
Vanilla + RandAugment	74.78	200.0
RtD	75.99	38.8

Controlling gradient magnitude. To test whether the benefit of strong augmentation comes from larger gradients alone, we use the norm matching control in Table 8. For each recycled sample, we retain the standard scoring input and multiply its cross entropy loss by the ratio of the strong input's logit gradient norm to the standard input's norm. We apply stop gradient (sg) to this ratio and cap the weight at 10^6. Table 16 shows that strong inputs outperform both norm matching and fresh standard inputs under both protocols. Table 17 confirms that mean magnitude realization reaches about 98% in all three seeds under Standard, yet the accuracy gap persists. These results support using strong augmentation to recycle excluded samples as a more effective way to supplement training than simply increasing their gradient magnitudes.

Table 16: Magnitude control on CIFAR-100 with EfficientNet-B0 over 200 epochs. Accuracy (%) is mean ± standard deviation over three seeds.

	Basic		Standard	
Recycled input	Acc. (%)	EE	Acc. (%)	EE
Norm matching	66.20 ± 0.18	31.3	75.34 ± 0.27	37.2
Fresh draw from $\mathcal{A}_{\text{std}}$	66.72 ± 0.22	30.4	76.16 ± 0.12	35.9
Fresh draw from $\mathcal{A}_{\text{strong}}$	$\mathbf{72.24} \pm 0.18$	32.1	$\mathbf{76.61} \pm 0.11$	36.9

Table 17: Norm matching over three seeds per protocol with cap 10^6. Mean realization averages epoch statistics.

	Basic		Standard	
Seed	Mean realization(%)	Capped(%)	Mean realization(%)	Capped(%)
A	88.5	10.1	98.3	1.7
B	89.5	9.3	97.9	2.0
C	89.8	9.1	98.1	1.9

Training before the final revision. Figure 5 complements Table 1 by showing how the accuracy advantage develops during training. On both datasets, RtD overtakes DBPD and Vanilla well before the final epoch and maintains higher accuracy thereafter. This sustained advantage shows that recycling benefits selective training throughout a substantial part of the schedule, rather than producing an improvement only at the final full data revision.

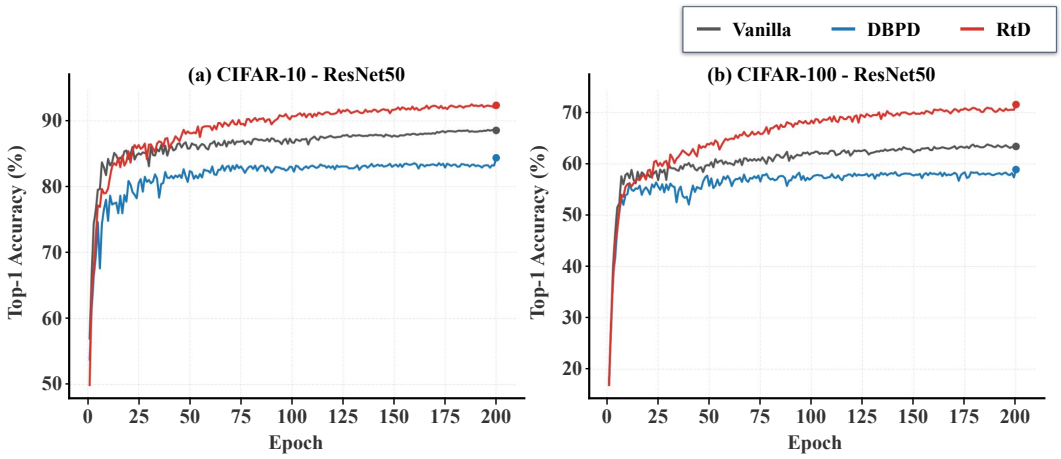


Figure 5: ResNet-50 test accuracy on CIFAR-10 (left) and CIFAR-100 (right) under the basic augmentation protocol. Full data revision is applied in the final epoch of the 200-epoch schedule.

E.2 Recycling Design and Sensitivity

Where to apply strong augmentation. Table 18 compares applying strong augmentation to keep samples, random samples, or drop candidates at a fixed backpropagation budget. The random and drop candidate variants use the same number of strongly augmented samples, while the keep sample variant provides a reference. Drop candidates give the highest accuracy on both datasets, supporting the choice to revisit samples excluded by the current model. This tests augmentation placement with $r = 1$, complementing the candidate pool comparison at the default $r = 0.1$ in Table 9.

Table 18: Placement of strong augmentation with EfficientNet-B0 at 30 EE. CIFAR-10 uses one seed. CIFAR-100 reports mean ± std. over three seeds.

Strong augmentation	CIFAR-10	CIFAR-100
None	88.81	64.17 ± 0.15
Keep samples	90.07	69.90 ± 0.21
Random samples	91.64	71.71 ± 0.26
Drop candidates ($r = 1$)	**92.00**	**72.07** ± 0.17

How to select drop candidates. To test whether prioritizing candidates by confidence improves selection within the drop candidate pool, Table 19 compares uniform sampling with selecting the lowest or highest confidence candidates first. Uniform sampling gives the highest accuracy on both datasets at the same recycling budget. We therefore use uniform sampling as the default candidate selection strategy.

Table 19: Selection within drop candidates with EfficientNet-B0 over 30 epochs. Accuracy (%) uses one seed and the same recycling budget.

Recycling selection	CIFAR-10	CIFAR-100
Lowest confidence first	89.56	67.80
Uniform (default)	90.34	68.53
Highest confidence first	87.75	66.57

Choice of strong augmentation. To assess whether recycling gains depend on a specific augmentation policy, Table 20 compares three policies applied to recycled samples. All three improve accuracy over no recycling, with similar EE among the recycling variants. RandAugment achieves the highest accuracy on both datasets, so we use it as the default strong augmentation policy.

Table 20: Strong augmentation policies with EfficientNet-B0 over 30 epochs. Entries give accuracy (%) / EE.

Recycled augmentation	CIFAR-10	CIFAR-100
None (no recycling)	87.41 / 5.59	65.78 / 11.12
RandomErasing + Flip	89.23 / 7.05	67.62 / 12.06
TrivialAugment + Flip	90.25 / 7.11	67.95 / 12.08
RandAugment + Flip	**90.34** / 7.03	**68.53** / 12.04

Recycle ratio. Figure 6 examines how the recycle ratio affects accuracy and backpropagation cost. Increasing r improves final accuracy but requires more EE, while the additional accuracy gains diminish at larger ratios. We therefore use $r = 0.1$ as the default, retaining much of the accuracy improvement while reducing backpropagation by about 88% relative to full data training.

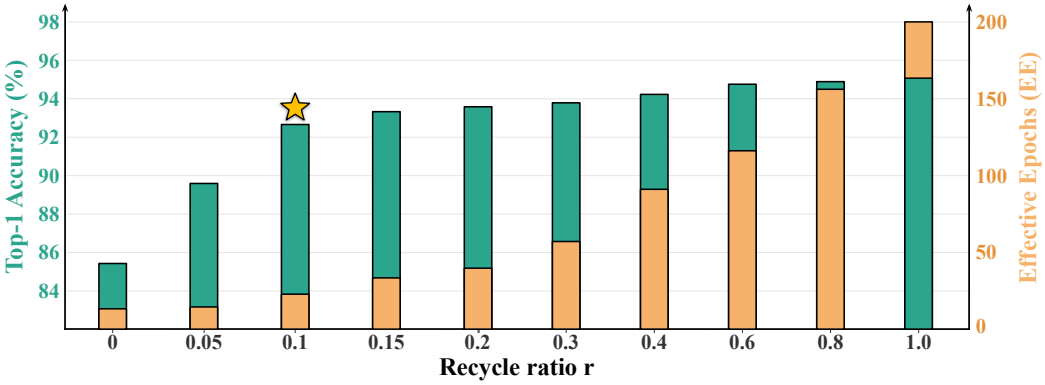


Figure 6: Recycle ratio on CIFAR-10 with EfficientNet-B0 over 200 epochs. Star denotes $r = 0.1$.

Confidence threshold. To assess sensitivity to the confidence threshold, Table 21 varies τ while holding the recycle ratio fixed. Increasing the threshold retains more samples for updates and therefore raises EE. Accuracy changes little from $\tau = 0.3$ to 0.9, so larger thresholds provide little additional benefit in this setting. We use $\tau = 0.3$ as the default, achieving the highest CIFAR-100 accuracy and remaining close to the best CIFAR-10 result with fewer updates overall.

Table 21: Comparison of confidence thresholds with EfficientNet-B0 over 30 epochs.

τ	CIFAR-10	CIFAR-100
0.1	88.49 / 5.6	67.74 / 9.4
0.3 (default)	90.34 / 7.0	**68.53** / 12.0
0.5	90.51 / 8.2	68.41 / 13.8
0.7	**90.59** / 9.3	68.39 / 15.4
0.9	90.44 / 11.3	68.24 / 18.0

E.3 Compatibility and Robustness

Recycling across pruning methods. Table 22 extends the standard protocol comparison in Table 5 to the basic protocol, applying recycling to InfoBatch, IES, and DBPD without changing their exclusion rules. Recycling with standard augmentation improves every baseline, and stronger augmentation improves each further. The same trend across both protocols supports the use of recycling with different pruning methods, although the gains come with additional backpropagation.

Table 22: Recycling across pruning methods on CIFAR-100 with EfficientNet-B0 for 200 epochs under the basic protocol.

Recycle	Strong aug.	InfoBatch	IES	DBPD
×	×	66.68/121.1	65.43/49.6	63.88/17.6
✓	×	66.92/128.9	66.50/62.9	66.72/30.4
✓	✓	69.90/129.9	70.09/88.6	72.24/31.7

Different optimizers. To evaluate recycling across optimizers, Table 23 compares four optimizers on both CIFAR datasets under the standard augmentation protocol, using the ResNet-50 settings in Appendix A. RtD and A-RtD achieve the two highest accuracies in every setting while reducing backpropagation by roughly 62% to 87% relative to Vanilla. These results show that the accuracy gains and backpropagation savings persist across the evaluated optimizers.

Table 23: ResNet-50 across different optimizers over 200 epochs.

CIFAR-10

Method	AdamW	SGD	LARS	LAMB
Vanilla	94.46 / 200.0	95.50 / 200.0	95.63 / 200.0	95.58 / 200.0
InfoBatch	94.49 / 118.7	95.53 / 129.6	95.83 / 124.2	95.59 / 123.5
OrderDP	94.16 / 127.5	95.07 / 127.5	95.60 / 127.5	95.47 / 127.5
RtD	**94.99** / 25.3	95.90 / 52.1	**96.26** / 34.4	**95.92** / 31.3
A-RtD	94.81 / 27.6	**96.06** / 61.6	96.11 / 38.1	95.86 / 34.3

CIFAR-100

Method	AdamW	SGD	LARS	LAMB
Vanilla	76.39 / 200.0	79.77 / 200.0	82.22 / 200.0	77.65 / 200.0
InfoBatch	75.81 / 124.1	79.79 / 134.8	81.98 / 135.0	78.72 / 135.0
OrderDP	76.14 / 127.5	78.89 / 127.5	82.19 / 127.5	78.98 / 127.5
RtD	**77.06** / 35.0	**80.46** / 58.3	82.71 / 69.1	**79.47** / 44.6
A-RtD	76.92 / 38.0	80.30 / 67.4	**82.90** / 75.2	79.31 / 48.8

Variation across seeds. To assess whether the gains persist across random seeds, Table 24 repeats the standard augmentation experiment over three seeds and 200 epochs. RtD and A-RtD exceed all reported pruning baselines in every seed while reducing backpropagation by about 80% relative to full data training. Both methods show small variation across seeds, supporting the consistency of the gains across the evaluated runs.

Table 24: Accuracy (%) over three seeds on CIFAR-100 with EfficientNet-B0 and standard augmentation. Mean reports mean ± standard deviation. EE is averaged over seeds.

Method	Seed A	Seed B	Seed C	Mean	EE
DBPD	74.34	74.78	73.87	74.33 ±0.46	24.7
SMRD	74.72	74.75	74.47	74.65 ±0.15	33.4
SRD	74.98	75.28	74.94	75.07 ±0.19	84.3
InfoBatch	74.51	74.95	74.32	74.59 ±0.32	124.1
RtD	76.63	76.71	76.49	**76.61** ±0.11	36.9
A-RtD	76.74	76.51	76.42	<u>76.56</u> ±0.17	39.5

E.4 Additional Data and Model Settings

Table 25: Fine grained classification with EfficientNet-B0 over 100 epochs.

Method	Aircraft	Flowers
Vanilla	80.32 / 100.0	95.66 / 100.0
DBPD	78.07 / 18.7	94.86 / 77.5
SRD	79.90 / 62.6	95.50 / 62.6
SMRD	79.15 / 15.0	88.68 / 9.6
InfoBatch	80.44 / 65.2	95.65 / 66.1
DivBS	80.41 / 50.0	95.62 / 50.0
IES	79.81 / 66.4	95.59 / 60.6
RtD	**81.16** / 18.5	<u>95.81</u> / 14.7
A-RtD	<u>80.84</u> / 19.7	**95.83** / 15.5

Table 26: Class imbalance with EfficientNet-B0 over 100 epochs.

Method	CIFAR-10-LT	CIFAR-100-LT
Vanilla	46.81 / 100.0	25.15 / 100.0
DBPD	52.16 / 8.2	25.97 / 19.9
SRD	48.17 / 63.0	26.90 / 63.0
SMRD	55.78 / 10.7	26.32 / 23.6
InfoBatch	50.53 / 61.0	26.90 / 65.4
DivBS	50.66 / 50.0	25.89 / 50.0
IES	47.83 / 65.9	25.65 / 89.2
RtD	<u>59.78</u> / 15.8	<u>30.45</u> / 28.3
A-RtD	**61.08** / 16.5	**30.47** / 29.6

Fine grained classification. To evaluate recycling across classification tasks, Table 25 compares methods on FGVC-Aircraft (Maji et al., 2013) and Oxford Flowers-102 (Nilsback & Zisserman, 2008). We use ImageNet pretrained EfficientNet-B0 with the augmentation settings in Appendix A. RtD and A-RtD achieve the two highest accuracies on both datasets while reducing backpropagation by more than 80% relative to Vanilla. These results extend the benefits of recycling beyond training from scratch to adapting pretrained models for fine grained classification.

Imbalanced classification. To assess whether these benefits persist when training samples are unevenly distributed across classes, Table 26 compares methods on CIFAR-10-LT and CIFAR-100-LT with an imbalance ratio of 100. RtD and A-RtD outperform all reported baselines on both datasets while reducing backpropagation by roughly 70% to 84% relative to Vanilla. A-RtD achieves the highest accuracy on both datasets, showing that recycling with decision reuse remains effective under class imbalance. Together, these experiments show that RtD and A-RtD improve accuracy while reducing backpropagation in both fine grained classification and imbalanced training.

Additional Transformer settings. To assess whether recycling benefits extend to Transformer backbones, Table 27 evaluates both fine tuning and training from scratch. We fine tune an ImageNet-1K MAE pretrained ViT-B (He et al., 2022) on CIFAR-100 under the basic protocol and train TinyViT (Wu et al., 2022) from scratch on CIFAR-10 under the standard protocol. RtD and A-RtD achieve the two highest accuracies in both settings while reducing backpropagation by roughly 86% to 88% relative to Vanilla. These results show that recycling and decision reuse improve accuracy while reducing backpropagation across the evaluated architectures and training settings.

Table 27: Additional Transformer settings over 200 epochs. Entries give accuracy (%) / EE.

Method	ViT-B CIFAR-100	TinyViT CIFAR-10
Vanilla	87.99 / 200.0	94.15 / 200.0
DBPD	87.09 / 12.3	90.95 / 8.5
SRD	87.35 / 84.3	93.90 / 84.3
SMRD	86.30 / 16.1	92.27 / 16.2
InfoBatch	88.15 / 117.3	93.77 / 118.1
DivBS	87.76 / 100.0	94.00 / 100.0
IES	87.33 / 12.3	93.82 / 51.6
RtD	<u>88.61</u> / 25.5	**94.58** / 24.8
A-RtD	**89.06** / 28.9	**94.71** / 27.6