EPOCH PRUNING: ACCELERATING MODEL TRAINING VIA GRADIENT REPLAY

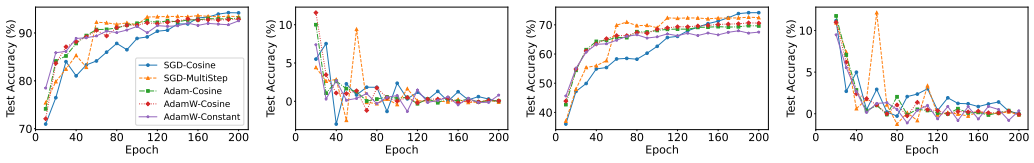
Anonymous authors
Paper under double-blind review

ABSTRACT

Training deep neural networks typically requires a prolonged training process. However, as training proceeds and the network gradually converges, the rate of performance improvement tends to diminish. While performance growth slows down, the training overhead stays the same. Thus, a natural question arises: can we adopt a differentiated strategy for the later stages of training, accelerating training and saving resources while maintaining effectiveness? Motivated by this, we introduce a novel concept termed *Epoch Pruning*. By reusing previously computed gradients, we can bypass the computationally expensive forward and backward propagation of certain later-stage epochs and update the model directly. Based on gradient replay, we devise two effective strategies to preserve the validity of the reused gradients. Extensive experiments conducted on multiple datasets, under both supervised and unsupervised learning settings, demonstrate that the proposed methods achieve approximately 1.2× to 1.4× speedup while maintaining the original model performance. The code is available in the *Supplementary Material*.

1 INTRODUCTION

As model parameter counts and dataset sizes continue to grow, the time required for model training has increased substantially, rendering methods for accelerating model training ever more essential (Menghani, 2023; Shen et al., 2024; Kaplan et al., 2020; Hestness et al., 2017; Sorscher et al., 2022; Yuan et al., 2025). However, the improvement in model performance does not increase linearly as training progresses. Rather, the model often exhibits substantial performance gains at the initial stages, while in later stages, as the model converges, the rate of performance improvement decelerates notably. To verify this, we employ ResNet-18 (He et al., 2016) to train the models using cross-entropy loss on CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009). A variety of representative optimizer and scheduler combinations are employed to ensure that the observed phenomena are generalizable. We adopt SGD with momentum (Sutskever et al., 2013) combined with cosine annealing (Loshchilov & Hutter, 2017) or multi-step scheduling, Adam (Kingma & Ba, 2015) with cosine annealing, and AdamW (Loshchilov & Hutter, 2019) with cosine annealing or a constant learning rate. Additional experimental details remain consistent and are provided in the Appendix A. Following common practice (Zagoruyko & Komodakis, 2016; Zhang et al., 2018; Yuan et al.,



(a) Model performance on CIFAR-10. (b) Performance gain on CIFAR-10. (c) Model performance on CIFAR-100. (d) Performance gain on CIFAR-100.

Figure 1: The performance and performance improvement of the model on the CIFAR-10 and CIFAR-100 datasets under different optimizer and scheduler configurations. It can be observed that the performance improvement is pronounced in the early stage of training, while it slows down in the later stage of training.

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.

2025; Zhang et al., 2024; Cho et al., 2025), we train the models for 200 epochs. Test set performance is recorded every 10 epochs, and the performance gain relative to the previous evaluation is computed. As shown in Figure 1, in the early stage of training, model performance improves more rapidly, whereas in the later stage, as the model gradually converges, the performance gains from training diminish progressively. Even when a constant learning rate is used, the performance gain in the later stage of model training remains inferior to that in the early stage, demonstrating that the difference in training improvement between the early and later stages is independent of commonly used learning rate decay strategies and results from the model having been sufficiently trained.

However, although the gains from training differ, the time and resources consumed during the early and later stages of training remain the same. Based on this imbalance, we should compress the time for late-stage training to a greater extent than early-stage training. Therefore, whether we can treat the early and later stages of model training differently to accelerate later-stage training without sacrificing performance deserves further research.

Inspired by network pruning (Han et al., 2016; Zhu et al., 2025; Frankle & Carbin, 2019), which reduces model size by removing non-critical parameters along the spatial dimension, and data pruning (Sorscher et al., 2022; Paul et al., 2021; Cho et al., 2025; Zhang et al., 2024), which reduces dataset size by discarding unimportant data points along the sample dimension, we propose a novel concept termed **Epoch Pruning**. This approach skips the time-consuming forward and backward propagation of certain training epochs, particularly during the later stages of model training. By effectively pruning away those epochs, we accelerate model training along the temporal dimension. However, this does not imply that the model weights cease to be updated. Instead, we can replay the gradients computed from previous training epochs as an estimate for gradient updates when propagation is omitted in the current epoch, thereby achieving resource savings while preserving the effectiveness of the training process.

While gradient replay offers a promising conceptual pathway to Epoch Pruning, realizing it in practice is non-trivial. A naive implementation that stores gradients from previous epochs per batch for one-to-one replay would face critical challenges. First, caching parameter gradients at every iteration introduces a severe memory bottleneck (Chaudhry et al., 2019). Even for a lightweight network like ResNet-18 (He et al., 2016), storing the gradients in FP32 for a single training step requires approximately 46.8 MB (about 11.7M parameters $\times$ 4 bytes). The storage overhead for an entire epoch's training trajectory grows linearly with the number of batches, rendering this approach unscalable and prohibitively expensive. Second, gradients generated during standard training can be noisy or biased (Keskar et al., 2017), so directly replaying them may amplify optimization errors.

To address above issues, we propose two specific and effective Epoch Pruning strategies: **Single-step Gradient Replay (SGR)**, which replays gradients by referring only to the previous epoch, and **Momentum Gradient Replay (MGR)**, which replays gradients by performing momentum updates using historical gradients from multiple epochs. Through theoretical analysis, both methods are guaranteed to have robust expected descent lower bounds. Both strategies perform well across multiple experimental scenarios on different datasets, maintaining the original performance while accelerating the model.

In summary, this work makes the following key contributions:

- Based on the inherent imbalance in performance growth between the early and later stages of the model, we pioneer the introduction of a new concept called Epoch Pruning, which aims to accelerate model training at the epoch level.

- Based on gradient replay, we propose two specific strategies, named Single-step Gradient Replay (SGR) and Momentum Gradient Replay (MGR). By effectively estimating the gradients required for weight updates, we save the forward and backward propagation time needed for the replay epochs.

- Experiments under different supervision regimes and optimization configurations across multiple datasets demonstrate the effectiveness of our proposed methods SGR and MGR, as well as the promise of our concept, Epoch Pruning.

2 Related Work

Pruning in Deep Learning. Network pruning assumes that existing networks are often over-parameterized, with only a small number of parameters participating in important decision-making (Han et al., 2015; Denil et al., 2013). By identifying important parameters, large networks can be compressed into smaller ones, thereby effectively saving computational resources and memory space (Han et al., 2016; He et al., 2017; Ma et al., 2023; Liu et al., 2017). On the other hand, data pruning assumes that there is a large amount of redundancy in the dataset (Toneva et al., 2019; Katharopoulos & Fleuret, 2018b). So a subset of samples can be selected based on their representativeness or importance, thereby directly accelerating model training while maintaining performance (Paul et al., 2021; Zhang et al., 2024; Cho et al., 2025; He et al., 2024). In contrast, Epoch Pruning is a novel pruning strategy that aims to prune epochs, preventing them from undergoing propagation, thereby accelerating model training.

Model Training Acceleration. Due to the increasing scale of both network architectures and datasets, model training time has grown accordingly (Shen et al., 2024; Menghani, 2023). To accelerate model training, a wide range of methods from various perspectives have been extensively proposed. In terms of data efficiency, researchers employ coreset selection and data pruning techniques to eliminate redundant samples, and leverage early training dynamics or proxy models to assign importance scores to individual samples (Sorscher et al., 2022; Paul et al., 2021; Toneva et al., 2019; Katharopoulos & Fleuret, 2018a; Pleiss et al., 2020; Coleman et al., 2020; Zheng et al., 2023). On the algorithmic and numerical optimization front, how to accelerate model training has been extensively studied, to name a few, learning rate scheduling (Goyal et al., 2017; Loshchilov & Hutter, 2017; Smith, 2017), large-batch training (Smith et al., 2018; Keskar et al., 2017), mixed-precision training (Micikevicius et al., 2018; Zhao et al., 2021), dynamic depth (Fan et al., 2020; Huang et al., 2016). Different from existing approaches, the proposed Epoch Pruning accelerates model training by directly bypassing the forward and backward passes of redundant epochs, while employing gradient replay to maintain comparable performance.

Gradient Replay. In continual learning, gradient replay techniques overcome catastrophic forgetting by projecting the update direction of a new task into a safe region defined by reference gradients of previous tasks (Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019; Chaudhry et al., 2019). Meanwhile, some studies treat the historical gradient as an anchor to correct the local noisy gradient produced by the current mini-batch data (Johnson & Zhang, 2013; Defazio et al., 2014). Although such methods rely on storing past gradient snapshots, during actual iterations the model still needs to perform forward propagation to obtain the instantaneous error under the current parameters, so as to accurately estimate the gradient correction required for optimal convergence in the current epoch (Nguyen et al., 2017; Gower et al., 2020).

3 Methodology

Based on gradient replay, we propose an effective Epoch Pruning paradigm. Specifically, we introduce two specific strategies, Single-step Gradient Replay (SGR) and Momentum Gradient Replay (MGR), so that the gradients used for replay are estimated more accurately.

3.1 Overview

When training a deep neural network (DNN), its parameters θ are optimized by minimizing a loss function $\mathcal{L}$ over a dataset $\mathcal{D}$. This process traditionally necessitates resource-intensive forward and backward passes at each iteration. Our approach aims to mitigate this computational burden while ensuring the continuous advancement of the model's optimization state. To achieve this, we introduce an alternating training mechanism that distinguishes between **Active Epoch** and **Replay Epoch**. During active epochs, we perform normal forward and backward propagation to update the model weights. In the replay epoch, instead of performing forward and backward propagation, we update the model weights through gradient replay. We use two hyperparameters to manage the switching between active epochs and replay epochs: E denotes the replay start epoch, after which gradient replay is allowed to commence. When the training epoch is less than E, replay is not permitted, and only normal training with active epochs is carried out; K denotes the replay interval, i.e., one replay epoch is performed every K epochs, while the other epochs remain as active epochs. In

our framework, we adopt an aggressive configuration by fixing $K = 2$. Consequently, the training strictly alternates between one active epoch and one replay epoch after the threshold E. Experiments validate that this approach yields superior performance. Replay start epoch E and the replay interval K determine the degree of model acceleration, specifically the total number of pruned epochs.

To replay gradients, it is necessary to retain the gradient information from previous steps. However, saving the gradients of all batches in each epoch for one-to-one replay is computationally expensive and memory-wise infeasible (Chaudhry et al., 2019). Therefore, we instead save, for each model parameter, the mean and standard deviation of its gradients across all batches in the epoch, as follows:

$$\mu = \frac{1}{B}\sum_{i=1}^{B} g_i, \quad \mu_{sq} = \frac{1}{B}\sum_{i=1}^{B} g_i^2 \quad \sigma = \sqrt{\mu_{sq} - \mu^2}, \tag{1}$$

where g_i is the gradient of the i_{th} batch, μ and μ_{sq} are the mean of gradients and the mean of squared gradients over the epoch, σ is the standard deviation of the gradients, and B is the number of batches in the epoch. In this way, we can accumulate the gradients and the squared gradients batch by batch during training. Therefore, when computing the gradient mean and standard deviation for the epoch, we only need to maintain a memory footprint equivalent to that of a single batch of gradients and squared gradients, rather than recording the gradients of all batches, thereby effectively saving memory.

We leverage a Gaussian distribution to generate the replayed gradients. This is motivated by prior work (Smith & Le, 2018; Zhu et al., 2019; Mandt et al., 2017), which has shown that mini-batch gradients can be mathematically modeled as Gaussian random variables. Moreover, our empirical analysis in Appendix F demonstrates that, within the same epoch, the gradients of most parameters follow a Gaussian distribution. Using the mean and standard deviation of the gradients, we approximate the gradients in the replay epochs. Specifically, the synthetic gradient $\tilde{g}$ is drawn from a Gaussian distribution by scaling and shifting standard stochastic noise:

$$\tilde{g} = \mu + z \cdot \sigma, \quad \text{where} \quad z \sim \mathcal{N}(0, I), \tag{2}$$

where $\mathcal{N}(0, I)$ denotes a multivariate Gaussian distribution with a zero mean vector and an identity covariance matrix. However, since the replayed gradients are derived from previously computed true gradients, any existing noise or biased estimations in these historical gradients can be inadvertently amplified. To address this issue, we propose two strategies to mitigate this effect without introducing significant computational overhead.

3.2 Single-step Gradient Replay

By leveraging the gradient statistics strictly from the most recent preceding active epoch, we formulate the **Single-step Gradient Replay (SGR)** method. In accordance with the Robbins-Monro stochastic approximation conditions (Robbins & Monro, 1951), the learning rate in model optimization must asymptotically decay to zero, enabling the model to gradually converge over the course of training. As a result, historical gradient statistics become increasingly accurate surrogates for current gradients in later epochs. Thus, we can progressively increase the weight assigned to the replayed gradients as training proceeds, thereby reducing the impact of erroneous gradients. We introduce a Progressive Linear Weighting method. Specifically, we apply a linearly increasing relationship between the gradient weights and the training epochs:

$$w_{\text{sgr}} = (T_{\text{current}} - E)/(T_{\text{total}} - E), \tag{3}$$

where T_{current} and T_{total} are the current epoch number and the total number of epochs respectively. Then we can reformulate $\tilde{g}$ as below:

$$\tilde{g}_{\text{sgr}} = w_{\text{sgr}} \cdot \tilde{g}, \tag{4}$$

Grounded in the convergence dynamics of neural network optimization, we introduce a linearly increasing weighting scheme for the replayed gradients to serve as a soft-start mechanism. This strategy mitigates the amplification of stochastic gradient noise during the initial replay phases, thereby ensuring a seamless transition to computationally efficient updates as the optimization trajectory stabilizes. Following this, we bypass standard forward and backward propagation, directly applying the synthesized gradients to update the model parameters according to the scheduled learning rate:

$$\theta^{t+1} = \theta^{t} - \eta^{t} \cdot \tilde{g}^{t}_{\text{sgr}}, \tag{5}$$

where θ^{t} and η^{t} are the model parameters and learning rate at t step.

3.3 MOMENTUM GRADIENT REPLAY

Given that the gradient direction during model optimization should ideally remain stable, whereas single-step gradients are highly susceptible to stochastic noise, we propose a momentum-based method, named **Momentum Gradient Replay (MGR)**. This approach leverages historical gradients across multiple epochs to robustly estimate the replayed gradients. Specifically, we employ an Exponential Moving Average (EMA) update scheme. We continuously update the global running mean ($\bar{\mu}$) and the running mean of squared gradients ($\bar{\mu}_{sq}$) for each parameter.

$$\begin{aligned} \bar{\mu}^{\mathrm{t}} &= \beta \bar{\mu}^{\mathrm{t\text{-}1}} + (1-\beta)\mu^{\mathrm{t}}, \\ \bar{\mu}^{\mathrm{t}}_{sq} &= \beta \bar{\mu}^{\mathrm{t\text{-}1}}_{sq} + (1-\beta)\mu^{\mathrm{t}}_{sq}, \end{aligned} \tag{6}$$

where $\beta \in [0, 1)$ is the momentum coefficient. In our experiments, β is fixed as 0.9. In this way, we ensure that the replayed gradients encapsulate the aggregated information of historical gradients, thus alleviating the adverse effects of instantaneous gradient errors.

Similar to Eq. (2), the synthetic gradient $\tilde{g}_{\mathrm{mgr}}$ is computed from the global running mean ($\bar{\mu}$) and the running mean of squared gradients ($\bar{\mu}_{sq}$) as below:

$$\tilde{g}_{\mathrm{mgr}} = \bar{\mu} + z \cdot \bar{\sigma}, \quad \text{where} \quad \bar{\sigma} = \sqrt{\bar{\mu}_{sq} - \bar{\mu}^2}. \tag{7}$$

Then the computed gradients are replayed in the same way as Eq. (5):

$$\theta^{\mathrm{t}+1} = \theta^{\mathrm{t}} - \eta^{\mathrm{t}} \cdot \tilde{g}^{\mathrm{t}}_{\mathrm{mgr}}, \tag{8}$$

The complete workflow of Epoch Pruning strategies is shown in Algorithm 1. Prior to epoch E, training consists solely of standard active epochs. From epoch E onward, replay occurs every K epochs. Crucially, gradients are recorded only during the active epoch immediately preceding a replay, ensuring a strict one-to-one mapping between gradient recording and replay execution.

Algorithm 1 Epoch Pruning strategies: SGR and MGR

Require: Total epoch number T_{total}, replay start epoch E and replay interval K.

for $t = 1$ **to** T_{total} **do**
  if $t < E$ **then** ▷ Before enabling replay, only the active epoch occurs
    Train the model normally (forward, backward, optimizer step).
  else ▷ Enabling replay, the active epoch and the replay epoch alternate
    if $t \,\%\, K \neq 0$ or $t = E$ **then** ▷ Active epoch
      Train the model normally.
      if $(t+1) \,\%\, K = 0$ **then** ▷ Next epoch is a replay epoch
        Record gradient statistics (mean and mean squared) via Eq. (1).
        If using MGR: update the running mean via Eq. (6).
    if $t > E$ and $t \,\%\, K = 0$ **then** ▷ Replay epoch
      If using SGR: generate replayed gradients via Eq. (2, 4), update model via Eq. (5).
      If using MGR: generate replayed gradients via Eqs. (2, 7), update model via Eq. (8).

3.4 THEORETICAL ANALYSIS OF PROPOSED METHODS

In this section, we theoretically analyze the optimization dynamics of the proposed Epoch Pruning strategies, SGR and MGR. We demonstrate that during replay epochs, both methods effectively regulate synthetic gradient variance and guarantee robust expected descent lower bounds.

We assume the objective function $L(\theta)$ is continuously differentiable and L-smooth. As performance growth decelerates in the later stages of training, we assume the true gradient distribution becomes locally stationary. Thus, historical statistics collected from the preceding active epoch provide an approximately unbiased estimation of the current true gradient, defined as $\mu = \nabla L(\theta_t) + \xi$, where ξ represents the negligible gradient drift error bounded by $||\xi|| \leq \epsilon$ under a bounded gradient norm $\|\nabla L(\theta_t)\| \leq G$. To quantify the fluctuation of the synthetic gradients, we use the trace of the covariance matrix to evaluate the overall variance penalty, denoted as $V = Tr(\Sigma) = \sum \sigma_i^2$.

Let $\Delta_t = L(\theta_t) - \mathbb{E}[L(\theta_{t+1})]$ denote the expected one-step descent. A larger lower bound on Δ_t indicates a more stable optimization step.

Theorem 1 *(Descent Lower Bound of SGR). Under above assumptions, updating the model using SGR with a progressive linear weight $w_{sgr} \in [0, 1]$ yields the expected descent bound:*

$$\Delta_{sgr} \geq \eta w_{sgr} \left(1 - \frac{L\eta w_{sgr}}{2}\right) \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2 w_{sgr}^2}{2} V_{t-1} - \mathcal{O}(\epsilon). \tag{9}$$

Theorem 2 *(Descent Lower Bound of MGR). Under above assumptions, updating the model using Momentum Gradient Replay (MGR) with EMA-smoothed historical gradients yields the expected descent bound, where $\bar{V}_t$ is the EMA-smoothed variance trace:*

$$\Delta_{mgr} \geq \eta \left(1 - \frac{L\eta}{2}\right) \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2}{2} \bar{V}_t - \mathcal{O}(\epsilon). \tag{10}$$

Due to space constraints, the detailed mathematical proofs and discussions for Theorems 1 and 2 are provided in Appendix H.

4 Experiments

In this section, we conduct extensive experiments to validate the effectiveness of the proposed Epoch Pruning methods, namely SGR and MGR. Specifically, our comprehensive evaluation encompasses supervised learning under various optimization configurations (Section 4.1), diverse unsupervised algorithms (Section 4.2), different network architectures (Section 4.3), and textual datasets (Section 4.4). Furthermore, we investigate their integration with data pruning training paradigms (Section 4.5), compare performance with early stopping (Appendix D), conduct thorough ablation studies (Section 4.6) and hyperparameter analysis (Appendix C), and provide a detailed analysis of speedup (Appendix E).

In our experiments, we evaluate the performance of the methods from two dimensions: top-1 prediction accuracy and speedup. Since Epoch Pruning represents a novel concept and problem setting, for fairness, we restrict the performance comparison of the proposed SGR and MGR to baselines without Epoch Pruning. In terms of accuracy, consistent with prior works on training acceleration (Micikevicius et al., 2018; Yuan et al., 2025; Paul et al., 2021), our objective is not to surpass the baseline performance but to approach it as closely as possible. Because training times vary with machines, experimental environments, and experimental methods, measuring the degree of speedup in absolute time is not readily transferable. Therefore, across different experiments, we qualitatively analyze the speedup ratio, and in Appendix E, we provide a detailed account of how the speedup ratio is calculated and determined, along with quantitative timing results and analyses for several methods. Specifically, we empirically demonstrate that the proposed methods introduce almost negligible overhead while substantially reducing standard training cost in Appendix E.

4.1 Supervised Learning under Various Optimization Configurations

To validate the effectiveness of the proposed method, we conduct experiments on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) using different optimizer settings. Consistent with Figure 1, we test five different configurations, including SGD with momentum (Sutskever et al., 2013) combined with cosine annealing (Loshchilov & Hutter, 2017) or multi-step scheduling, Adam (Kingma & Ba, 2015) with cosine annealing, and AdamW (Loshchilov & Hutter, 2019) with cosine annealing or a constant learning rate. We use ResNet-18 as the backbone and utilize cross-entropy loss. Following (Zagoruyko & Komodakis, 2016; Zhang et al., 2018; Yuan et al., 2025; Zhang et al., 2024; Cho et al., 2025), we train the model for 200 epochs. Other details are listed in Appendix A. In SGR and MGR, we set the replay start epoch E to 100 (half of the total training epochs). As illustrated in the Section 3, the replay interval K and the momentum coefficient β are set to 2 and 0.9, respectively, in all experiments unless otherwise specified. As shown in Table 1, the proposed methods achieve comparable performance to the baseline that is normally trained without Epoch Pruning, using the same settings. The proposed SGR or MGR even slightly outperforms the baseline in some cases, where results are bolded. In other cases, the performance of Epoch Pruning is close to that without

pruning. Because we start Epoch Pruning from the halfway point of the training process, a speedup of approximately 1.3× is achieved. A detailed calculation of the speedup ratio is given in Appendix E. In addition, we present the accuracy curves over training epochs in Appendix G to verify that the proposed methods can closely fit the training process of the No Pruning baseline and achieve performance convergence.

Table 1: Performance (in percentage) of Epoch Pruning on CIFAR-10 and CIFAR-100 under multiple optimization configurations. The cases that outperform the baseline are highlighted in bold, and a speedup of approximately 1.3× is achieved.

	CIFAR-10			CIFAR-100		
Configuration	No Pruning	SGR	MGR	No Pruning	SGR	MGR
SGD-Cosine	94.30 ± 0.07	94.19 ± 0.12	94.16 ± 0.14	74.03 ± 0.13	73.99 ± 0.29	$\mathbf{74.18 \pm 0.35}$
SGD-MultiStep	93.37 ± 0.09	$\mathbf{93.44 \pm 0.12}$	$\mathbf{93.53 \pm 0.14}$	72.77 ± 0.31	72.37 ± 0.27	72.74 ± 0.41
Adam-Cosine	93.01 ± 0.04	92.98 ± 0.22	$\mathbf{93.13 \pm 0.19}$	69.71 ± 0.10	$\mathbf{70.01 \pm 0.04}$	$\mathbf{69.99 \pm 0.36}$
AdamW-Cosine	93.03 ± 0.17	$\mathbf{93.08 \pm 0.24}$	$\mathbf{93.05 \pm 0.07}$	70.37 ± 0.33	70.28 ± 0.57	70.05 ± 0.06
AdamW-Constant	92.03 ± 0.37	91.11 ± 0.73	91.91 ± 0.24	67.03 ± 0.35	66.84 ± 0.24	$\mathbf{67.16 \pm 0.41}$

Table 2: Performance (in percentage) of Epoch Pruning across ImageNet and Tiny-ImageNet.

Method	Tiny-ImageNet		ImageNet	
	SGD-Cosine	SGD-MultiStep	SGD-Cosine	SGD-MultiStep
No Pruning	63.95 ± 0.37	60.97 ± 0.16	72.99 ± 0.09	70.67 ± 0.08
SGR	63.80 ± 0.21	61.30 ± 0.27	71.52 ± 0.22	69.28 ± 0.18
MGR	63.70 ± 0.50	61.31 ± 0.16	71.50 ± 0.31	70.39 ± 0.22

We also conduct experiments on Tiny-ImageNet (Le & Yang, 2015) and the large-scale dataset ImageNet-1K (Deng et al., 2009), which contains 200 and 1,000 classes respectively. Since training on large datasets is more time-consuming, the need for effective training acceleration methods becomes even more critical. For our experimental setup, we utilize ResNet-50 on ImageNet-1K and ResNet-34 on Tiny-ImageNet as the backbone networks. All models undergo training for 90 epochs, optimized via SGD with momentum alongside either a cosine annealing or a multi-step learning rate scheduler. We set replay start epoch E to 45, which indicates an approximately 1.3× speedup (detailed in Appendix E). As shown in Table 2, the proposed methods achieve comparable performance while saving training time.

4.2 Unsupervised Learning across Various Frameworks

Compared with supervised learning, unsupervised learning typically requires a larger number of epochs to achieve comparable performance (Chen et al., 2020). Based on Epoch Pruning, our proposed SGR and MGR can effectively accelerate unsupervised training. Given the significant differences among various unsupervised training frameworks, we selected five well-known and representative methods, to validate the effectiveness of proposed methods. Specifically, we evaluate SimCLR (Chen et al., 2020), which relies on negative samples for contrastive learning; BYOL (Grill et al., 2020), a non-contrastive approach utilizing an EMA without negative samples; SimSiam (Chen & He, 2021), a non-contrastive method requiring neither negative samples nor EMA; MoCo (He et al., 2020), a contrastive framework that maintains a memory queue to supply negative samples; and Barlow Twins (Zbontar et al., 2021), which is grounded in redundancy reduction. Collectively, these five methods encompass the predominant training methodologies currently utilized in unsupervised learning. We train all methods using ResNet-18 (He et al., 2016) as the backbones for 600 epochs, and perform Epoch Pruning starting from $E = 400$, which corresponds to an approximate speedup of 1.2×. All models are trained using SGD with momentum and a cosine annealing scheduler, the full details of which are in Appendix A. We adopt top-1 accuracy of k-nearest neighbors (K-NN) and linear probing as the performance metrics. As shown in Table 3, under different unsupervised algorithms, the proposed methods achieve performance comparable to that of no pruning while substantially reducing training time.

Table 3: Performance (in percentage) of Epoch Pruning for unsupervised learning. A speedup of approximately 1.2× is achieved.

Dataset	Method	K-NN Accuracy			Linear Probing Accuracy		
		No Pruning	SGR	MGR	No Pruning	SGR	MGR
CIFAR-10	SimCLR	85.83 ± 0.06	85.34 ± 0.18	85.35 ± 0.12	88.69 ± 0.15	88.29 ± 0.12	88.39 ± 0.11
	BYOL	85.41 ± 0.18	85.18 ± 0.29	85.25 ± 0.09	86.40 ± 0.21	85.86 ± 0.16	86.39 ± 0.29
	SimSiam	86.86 ± 0.12	85.35 ± 0.18	86.40 ± 0.20	88.29 ± 0.12	88.29 ± 0.13	88.12 ± 0.14
	MoCo	85.89 ± 0.20	85.36 ± 0.25	85.41 ± 0.13	87.63 ± 0.19	87.04 ± 0.20	86.97 ± 0.05
	Barlow Twins	85.30 ± 0.16	84.64 ± 0.41	85.06 ± 0.21	87.60 ± 0.36	86.64 ± 0.19	87.32 ± 0.44
CIFAR-100	SimCLR	49.61 ± 0.19	48.91 ± 0.13	48.84 ± 0.22	57.30 ± 0.29	57.02 ± 0.26	56.96 ± 0.31
	BYOL	54.29 ± 0.56	53.41 ± 0.12	53.78 ± 0.17	54.70 ± 0.18	54.38 ± 0.16	55.10 ± 0.10
	SimSiam	52.16 ± 0.30	51.18 ± 0.17	51.17 ± 0.35	54.55 ± 0.31	53.45 ± 0.18	53.21 ± 0.13
	MoCo	54.44 ± 0.16	53.79 ± 0.26	53.59 ± 0.08	58.21 ± 0.19	57.47 ± 0.21	57.43 ± 0.37
	Barlow Twins	50.05 ± 0.40	49.38 ± 0.49	49.52 ± 0.52	59.34 ± 0.45	58.14 ± 0.61	57.96 ± 0.59

4.3 Evaluation across Different Network Architectures

We validate the robustness of our methods across multiple widely-used backbones, including ResNet-18, ResNet-50 (He et al., 2016), VGG-16 (Simonyan & Zisserman, 2015), MobileNetV2 (Sandler et al., 2018) and ViT-Tiny (Dosovitskiy et al., 2021; Touvron et al., 2021). We use SGD with momentum for all models, except for the transformer-based model ViT-Tiny, which is optimized using AdamW. A cosine annealing learning rate scheduler is applied across all evaluated models. Other settings are consistent with Section 4.1. As shown in Table 4, across different model architectures, the two proposed Epoch Pruning strategies achieve comparable performance while effectively accelerating model training.

Table 4: Performance (in percentage) of Epoch Pruning across various network architectures.

Model	CIFAR-10			CIFAR-100		
	No Pruning	SGR	MGR	No Pruning	SGR	MGR
ResNet-18	94.30 ± 0.07	94.19 ± 0.12	94.16 ± 0.14	74.03 ± 0.13	73.99 ± 0.29	74.18 ± 0.35
ResNet-50	94.94 ± 0.21	94.97 ± 0.10	94.96 ± 0.04	77.03 ± 0.26	77.32 ± 0.29	77.07 ± 0.15
VGG-16	93.24 ± 0.07	93.11 ± 0.18	93.06 ± 0.25	71.47 ± 0.11	71.57 ± 0.30	71.35 ± 0.16
MobileNetV2	90.34 ± 0.06	89.34 ± 0.05	89.63 ± 0.16	66.83 ± 0.15	66.47 ± 0.09	66.42 ± 0.19
ViT-Tiny	83.98 ± 0.07	83.97 ± 0.07	84.00 ± 0.29	53.40 ± 0.37	53.43 ± 0.47	53.63 ± 0.57

4.4 Evaluation on Text Datasets

Since our primary experiments focus on image datasets, to verify the applicability to another modality, we further evaluate our methods on text datasets, which represents another major modality besides images. Specifically, we employ TextCNN (Kim, 2014) as the backbone, training and testing on AG News (4-class) and DBpedia (14-class) (Zhang et al., 2015). We apply SGD-Cosine configuration and train each model for 20 epochs. We start enabling replay at the half of training progress, i.e., E=10. In this way, a speedup of about 1.3× is achieved. Additional training specifics are detailed in Appendix A. The performance is shown in Table 5. The performance results demonstrate the applicability of the proposed SGR and MGR strategies across different modalities.

Table 5: Performance (in percentage) of Epoch Pruning on text datasets. A speedup of approximately 1.3× is achieved.

	AG News	DBpedia
No Pruning	89.40±0.20	98.05±0.03
SGR	89.02±0.19	98.00±0.01
MGR	89.04±0.13	97.99±0.03

4.5 Integration with Data Pruning Methods

As an effective method for accelerating model training, Epoch Pruning can potentially be integrated with other acceleration techniques. Data pruning accelerates model training at the data level by sampling valuable instances from the dataset. Meanwhile, our proposed Epoch Pruning accelerates training at the temporal level by dis-

carding the propagations of later-stage epochs. Since these two paradigms are orthogonal, we investigate whether their combination can further accelerate the model training. We combine Epoch Pruning with four recent state-of-the-art data pruning methods to investigate whether their mechanisms conflict or complement each other. We select EL2N (Paul et al., 2021), TDDS (Zhang et al., 2024), Dyn-Unc (He et al., 2024) and DUAL (Cho et al., 2025) for evaluation. We conduct standard experiments for 200 epochs on CIFAR-10 and CIFAR-100, using SGD-Cosine configuration. At the 30-th epoch, based on the output probabilities from the preceding 30 epochs, we prune 30% of redundant samples according to their respective algorithms. Concurrently, we apply Epoch Pruning from the 100-th epoch (i.e., setting $E = 100$). The results are presented in Table 6. As observed, the integration of data pruning methods with our proposed Epoch Pruning strategies further accelerates training. Moreover, applying Epoch Pruning on top of data pruning does not lead to a significant performance drop, yielding comparable results. Also, the final performance is predominantly determined by the data pruning baseline. This further demonstrates the applicability and robustness of proposed Epoch Pruning strategies, namely SGR and MGR.

Table 6: Performance (%) of integrating Epoch Pruning with data pruning on CIFAR.

Dataset	Training Setting	EL2N	TDDS	Dyn-Unc	DUAL	Speedup
CIFAR-10	Only Data Pruning	94.05 ± 0.23	92.98 ± 0.03	91.94 ± 0.06	94.07 ± 0.04	1.3×
	w/ SGR	93.82 ± 0.11	92.97 ± 0.17	91.97 ± 0.07	94.00 ± 0.07	1.7×
	w/ MGR	93.94 ± 0.13	92.71 ± 0.19	91.85 ± 0.19	94.03 ± 0.16	1.7×
CIFAR-100	Only Data Pruning	69.66 ± 0.41	70.09 ± 0.52	67.62 ± 0.36	69.48 ± 0.51	1.3×
	w/ SGR	70.43 ± 0.22	69.90 ± 0.22	67.68 ± 0.23	69.00 ± 0.16	1.7×
	w/ MGR	70.78 ± 0.39	69.61 ± 0.12	67.86 ± 0.15	68.92 ± 0.13	1.7×

4.6 Ablation on Acceleration Mechanisms

We conduct ablation studies to validate the effectiveness of the proposed Epoch Pruning. We conduct evaluations using the SGD-Cosine configuration and the replay setup is consistent with that in Section 4.1, which means that once replay is enabled, half of the epochs are directly replayed, while the other half undergo normal training. Therefore, we employ no-pruning training with early stopping at 50% and 75% of the total training progress as our comparison methods (denoted as ES (50%) and ES (75%), respectively). In this way, we can demonstrate that the model has not yet attained its final performance before gradient replay, and that the performance continues to rise when gradient replay strategies are employed. We also test the performance without gradient replay, i.e., directly skipping the weight updates during the replay epochs, to demonstrate whether the gradients replayed by the proposed SGR and MGR strategies facilitate model training. The results are shown in Table 7. Evidently, relative to the comparison approaches, SGR and MGR yield results nearest to the No Pruning baseline, which validates that the proposed Epoch Pruning strategies can both effectively utilize gradient replay for performance improvement. Regarding the design of the PLW in SGR and the momentum in MGR, we present the rationale in Appendix B. Also, the impact of hyperparameters is studied in Appendix C.

Table 7: Ablation study (%) on the acceleration mechanisms.

Method	CIFAR-10	CIFAR-100
ES (50%)	88.19 ± 1.05	63.71 ± 0.58
ES (75%)	92.16 ± 0.44	70.59 ± 0.71
w/o Gradient Replay	93.91 ± 0.10	73.08 ± 0.31
SGR (Ours)	94.19 ± 0.12	73.99 ± 0.29
MGR (Ours)	94.16 ± 0.14	74.18 ± 0.35
No Pruning	94.30 ± 0.07	74.03 ± 0.13

5 Conclusion

During late-stage training, performance gains diminish while computational overhead remains constant, making training cost-ineffective. To accelerate training without sacrificing performance, we introduce a novel concept termed *Epoch Pruning*. In pruned late-stage epochs, forward and backward propagation are bypassed, and model weights are updated via gradient replay. To avoid storing massive actual gradients, we generate replayed gradients from historical means and standard deviations. We propose Single-step Gradient Replay (SGR) and Momentum Gradient Replay (MGR) as strategies to achieve reliable gradient replay. Extensive evaluations across diverse experimental settings demonstrate our methods' efficiency and effectiveness.

AI USE STATEMENT

In this work, we used generative AI tools solely for writing assistance, including grammar and spell checking, and improving sentence clarity and readability. We have not used generative AI tools for other tasks in this work. We have reviewed all AI-assisted work: all LLM-generated or LLM-edited text was manually checked by the authors. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The code for the proposed methods is provided in the supplementary material to facilitate reproducibility. In addition, we provide a detailed description of the experimental setup in Appendix A and in all experiment-specific sections.

REFERENCES

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pp. 1597–1607. PMLR, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

Yeseul Cho, Baekrok Shin, Changmin Kang, and Chulhee Yun. Lightweight dataset pruning without full training via example difficulty and prediction uncertainty. In *International Conference on Machine Learning*, pp. 10602–10643. PMLR, 2025.

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 *International Conference on Learning Representations*, 2020.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems*, 27, 2014.

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*, pp. 248–255, 2009.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in Neural Information Processing Systems*, 26, 2013.

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. In *International Conference on Learning Representations*, 2021.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Robert M Gower, Mark Schmidt, Francis Bach, and Peter Richtárik. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11):1968–1983, 2020.

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.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284. Curran Associates, Inc., 2020.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

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*, June 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.

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

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international Conference on Computer Vision*, pp. 1389–1397, 2017.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems*, 26, 2013.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International Conference on Machine Learning*, pp. 2525–2534. PMLR, 2018a.

Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2525–2534. PMLR, 10–15 Jul 2018b.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

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

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international Conference on Computer Vision*, pp. 2736–2744, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems*, 36:21702–21720, 2023.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12):1–37, 2023.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pp. 2613–2621. PMLR, 2017.

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.

Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056, 2020.

Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pp. 55–69. Springer, 2002.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

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.

Li Shen, Yan Sun, Zhiyuan Yu, Liang Ding, Xinmei Tian, and Dacheng Tao. On efficient training of large-scale deep learning models. *ACM Computing Surveys*, 57(3):1–36, 2024.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.

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.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pp. 1139–1147. pmlr, 2013.

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. In *International Conference on Learning Representations*, 2019.

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.

Suqin Yuan, Runqi Lin, Lei Feng, Bo Han, and Tongliang Liu. Instance-dependent early stopping. In *The Thirteenth International Conference on Learning Representations*, 2025.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Xin Zhang, Jiawei Du, Yunsong Li, Weiying Xie, and Joey Tianyi Zhou. Spanning training progress: Temporal dual-depth scoring (tdds) for enhanced dataset pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26223–26232, 2024.

Ruizhe Zhao, Brian Vogel, Tanvir Ahmed, and Wayne Luk. Reducing underflow in mixed precision training by gradient scaling. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 2922–2928, 2021.

Haizhong Zheng, Rui Liu, Fan Lai, and Atul Prakash. Coverage-centric coreset selection for high pruning rates. In *The Eleventh International Conference on Learning Representations*, 2023.

Kehan Zhu, Fuyi Hu, Yuanbing Ding, Wei Zhou, and Ruxin Wang. A comprehensive review of network pruning based on pruning granularity and pruning time perspectives. *Neurocomputing*, 626:129382, 2025.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7654–7663. PMLR, 09–15 Jun 2019.

A EXPERIMENTAL SETUP

In this section, we detail the experimental setup used in this paper. We also provide the code of proposed methods in the supplementary material.

Datasets. We mainly use the CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), Tiny-ImageNet (Le & Yang, 2015), and ImageNet (Deng et al., 2009) image datasets, which have 10, 100, 200, and 1000 classes, respectively. We train on the predefined training sets and evaluate performance on the test sets. In Section 4.4, we train and test models on AG News and DBpedia (Zhang et al., 2015), which are text datasets containing 4 and 14 classes respectively.

Network Architectures. We mainly use ResNet-18 (He et al., 2016) as the backbone, unless otherwise specified. For all experiments on CIFAR-10, CIFAR-100 and Tiny ImageNet, we set the first convolution layer to have a kernel size of 3×3 and a stride of 1, and remove the first max-pooling layer in ResNet due to relatively small image sizes. In Section 4.3, we examine the robustness of the proposed methods across various network architectures, including ResNet-18, ResNet-50 (He et al., 2016), VGG-16 (Simonyan & Zisserman, 2015), MobileNetV2 (Sandler et al., 2018) and ViT-Tiny (Dosovitskiy et al., 2021; Touvron et al., 2021). Similarly, on the CIFAR-10 and CIFAR-100 datasets, to accommodate the small image sizes, we modify the input dimension of the first linear layer of VGG-16 to 512, change the stride of the first convolutional layer of MobileNetV2 from 2 to 1, and set the patch size of ViT-Tiny to 4.

Batch Size. In all experiments, we adopt a batch size of 256, unless otherwise specified.

Epochs. Unless otherwise specified, the number of epochs for this experiment is chosen as follows. For supervised learning tasks on CIFAR-10 and CIFAR-100, we train models for 200 epochs, following common practice (Zagoruyko & Komodakis, 2016; Zhang et al., 2018; Yuan et al., 2025; Zhang et al., 2024; Cho et al., 2025). For supervised learning tasks on Tiny ImageNet and ImageNet-1K, we train models for 90 epochs, following (Krizhevsky et al., 2012; Zhang et al., 2018; 2024; Cho et al., 2025). For unsupervised learning on CIFAR-10 and CIFAR-100 in Section 4.2, we empirically adopt 600 epochs for all methods. For text classification tasks in Section 4.4, we empirically set the number of epochs to 20.

Hyperparameters for SGR and MGR. For all experiments related to SGR and MGR, unless otherwise specified, we set the replay interval $K = 2$ and MGR momentum $\beta = 0.9$. For supervised learning methods, unless otherwise specified, we set the replay start epoch E to half of the total training epochs. For the unsupervised learning methods adopted in this paper, the total training is 600 epochs, and we set $E = 400$.

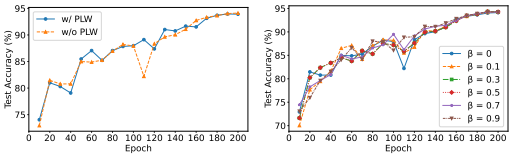
Optimizer and Scheduler Configurations. For supervised learning on image datasets, we employ multiple optimizer and scheduler configurations: (1) SGD-Cosine: using the SGD with momentum optimizer and the cosine annealing scheduler. (2) SGD-MultiStep: using the SGD with momentum optimizer and the multi-step scheduler. (3) Adam-Cosine: using the Adam optimizer and the cosine annealing scheduler. (4) AdamW-Cosine: using the AdamW optimizer and the cosine annealing scheduler. (5) AdamW-Constant: using the AdamW optimizer and a fixed constant learning rate. For SGD with momentum, the momentum, weight decay and initial learning rate are set to 0.9, 0.0005 and 0.1, respectively. For Adam and AdamW, weight decay is set to 0 and 0.01, respectively, and the initial learning rate is set to 0.001. For the cosine annealing scheduler, we apply linear warm-up for the first 5 and 10 epochs for experimental settings with total epochs of 90 and 200, respectively. For the multi-step scheduler, we decay the learning rate by a factor of 0.1 at milestones [50, 100] for 200-epoch experiments and at milestones [30, 60] for 90-epoch experiments. For supervised learning on text datasets, we apply SGD-cosine without linear warmup, where the momentum, weight decay and initial learning rate are set to 0.9, 0.00001 and 0.1. For unsupervised learning, we apply SGD-cosine with a linear warmup of 10 epochs, and set momentum, weight decay and initial learning rate to 0.9, 0.0001 and 0.05. Also, we employ Automatic Mixed Precision (AMP) (Micikevicius et al., 2018) during training models on ImageNet-1K.

Hyperparameters for Unsupervised Methods. For all methods, the representation dimension is set to 256, except for Barlow Twins, where it is set to 1024. In SimCLR (Chen et al., 2020), the temperature is set to 0.5. In BYOL (Grill et al., 2020), we set BYOL momentum to 0.996. For SimSiam (Chen & He, 2021), we set the learning rate of the predictor fixed at 0.05. In MoCo (He et al., 2020), the temperature, the queue size and MoCo momentum are set to 0.2, 4096 and

0.999. For Barlow Twins (Zbontar et al., 2021), we set the trade-off weight to 0.005. For the K-NN evaluation, we set the number of nearest neighbors to 200 and the temperature parameter to 0.1. For linear probing, we replace the final MLP of the original model with a linear layer, training only this layer for 100 epochs with a constant learning rate of 0.1.

B Design Rationale for Gradient Replay Strategies

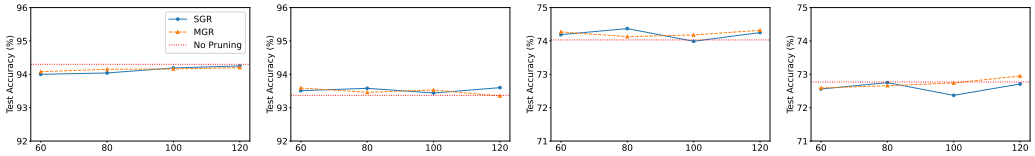
We further elaborate on the design rationale behind Progressive Linear Weighting (PLW) in SGR and investigate the impact of various momentum coefficients β in MGR. As shown in Figure 2, whether PLW is used in SGR and whether the momentum-based gradient update mechanism is used in MGR have little impact on the final model performance. This indicates that the gradient replay method itself is well suited to model training. However, at the onset of replay, without PLW or the momentum mechanism, the training progress exhibits fluctuations. To address this, we adopt PLW in SGR to mitigate the impact of early-stage replay on training stability, and employ the momentum mechanism in MGR to obtain smoother replayed gradients, thereby achieving more reasonable training progress. Specifically, as illustrated in Figure 2a, the presence or absence of Progressive Linear Weighting in SGR significantly influences the extent of performance degradation. As depicted in Figure 2b, higher momentum coefficients (e.g., 0.9) are preferable for MGR, as smaller coefficients tend to exacerbate the drop in performance. Therefore, we conclude that employing Progressive Linear Weighting for SGR and a high momentum coefficient for MGR provides a more robust and stable configuration for the proposed methods.



(a) SGR (b) MGR

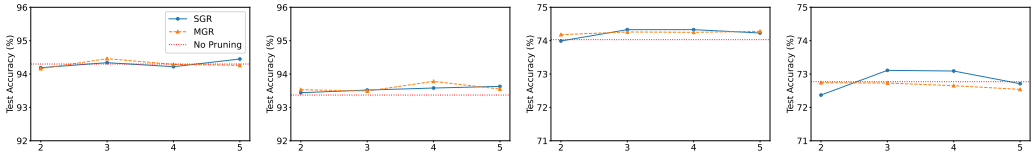
Figure 2: (a) Comparison of SGR with and without Progressive Linear Weighting (PLW); (b) Impact of different momentum coefficients on MGR. The results demonstrate that both Progressive Linear Weighting in SGR and the momentum mechanism in MGR are effective approaches for enhancing algorithmic stability.

C Hyperparameter Analysis



(a) CIFAR-10, SGD-Cosine (b) CIFAR-10, SGD-MultiStep (c) CIFAR-100, SGD-Cosine (d) CIFAR-100, SGD-MultiStep

Figure 3: The impact of replay start epoch E on model performance.



(a) CIFAR-10, SGD-Cosine (b) CIFAR-10, SGD-MultiStep (c) CIFAR-100, SGD-Cosine (d) CIFAR-100, SGD-MultiStep

Figure 4: The impact of replay interval K on model performance.

In the proposed Epoch Pruning method, the acceleration rate is governed by two key parameters: the replay start epoch E and the replay interval K. In this section, we investigate the impact of these two parameters.

To better investigate the effect of replay start timing on model performance, we conduct experiments on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) using a 200-epoch training setup and vary the replay start epoch E. This 200-epoch setting is a standard configuration adopted by many methods (Zagoruyko & Komodakis, 2016; Zhang et al., 2018; Yuan et al., 2025; Zhang et al., 2024; Cho et al., 2025) to sufficiently train the model, and it is also the setting used in this paper. We adopt two optimizer and scheduler configurations, including SGD-Cosine and SGD-MultiStep. As illustrated in Figure 3, even if we use an earlier replay start epoch E, the model can still achieve good performance comparable to the No Pruning baseline. However, since fewer epochs are actually trained, this saves more time and thus improves speedup ratio. In this case, when the replay start epoch E is set to 60, the model achieves approximately 1.4× speedup with almost no performance loss. In this paper, our primary focus is to demonstrate the concept of Epoch Pruning and the effectiveness of the proposed gradient replay methods, SGR and MGR. Therefore, we uniformly set the replay start epoch to $E = 100$ in Section 4.1, rather than meticulously tuning for a setting that maximizes time savings without performance loss. Using the same setup, we evaluated the impact of the replay interval K on model performance. In the proposed method, K is set to 2, which is the most time-efficient setting for training. We find that increasing K is unlikely to yield significant performance gains, as shown in Figure 4. Therefore, we prefer the replay interval $K = 2$ in our experiments.

D Comparison with Early Stopping

To validate the superiority of the proposed method, we compare it with the commonly used early stopping based on validation loss (Prechelt, 2002). Due to the different experimental settings, we conduct this comparison separately here. We adopt ResNet-18 as the backbone and use the SGD-Cosine configuration. We use half of the test set as the validation set to compute the loss, and early stopping is triggered when the loss does not decrease for a certain number of consecutive epochs. We empirically set this number to 15. We compare the performance of SGR and MGR using the same number of actual training epochs. Specifically, based on the number of epochs at early stopping, we calculate the total number of epochs required when Epoch Pruning starts at the halfway point of training, thereby ensuring a fair comparison. We evaluate performance on the full test set. As shown in Table 8, the proposed methods outperform early stopping. Compared to early stopping, Epoch Pruning essentially sustains the training process, merely bypassing the computationally expensive forward and backward passes. Given the same number of actual training epochs, Epoch Pruning enables more thorough training, thereby ensuring continuous performance gains.

Table 8: Performance comparison between the proposed Epoch Pruning strategy and early stopping with the same numbers of training epochs

	CIFAR-10	CIFAR-100
early stopping	87.33±1.18	67.48±1.79
SGR	92.41±0.18	72.88±0.23
MGR	92.31±0.10	72.83±0.07

E Speedup Analysis

We evaluate the time overhead of the proposed Epoch Pruning strategies, SGR and MGR, to further demonstrate their efficiency. We trained a ResNet-18 model on the CIFAR-10 dataset using a single NVIDIA GeForce RTX 3090 GPU. We adopt several representative methods, including standard supervised learning with Cross-Entropy loss, as well as self-supervised approaches like SimCLR and BYOL. Notably, standard supervised learning with Cross-Entropy loss and SimCLR share the same network architecture while BYOL adopts an asymmetric network architecture featuring two ResNet-18 backbones. The conventional training process is partitioned into several stages, including data loading, forward propagation, backward propagation and model update (via optimizer). For SGR and MGR, we specifically account for the time spent on gradient replay and gradient recording, respectively. Gradient replay accounts for the time spent generating gradients based on the mean and standard deviation, as well as the subsequent model update. Gradient recording encompasses the duration of collecting gradient statistics per batch and computing the mean and standard deviation. We calculate the mean time for each component averaged over 50 executions, and results are shown in Table 9. Gradient recording takes place only in the active epochs after the replay start epoch E, concurrently with normal training, whereas gradient replay is exclusively executed during replay

epochs. Notably, both operations require only about one second per epoch. They do not introduce substantial computational overhead, but Epoch Pruning can save significant time otherwise spent on normal training. Although Cross-Entropy training and SimCLR share the same dataset and network architecture, their normal training time overheads differ due to differences in algorithmic complexity, whereas the time required for gradient replay remains consistent. Because of their different network architectures, BYOL and the other two methods have different overheads for both normal training and Epoch Pruning, but the difference in normal training overhead is clearly larger.

Table 9: Training time consumption breakdown (per epoch) for Cross-Entropy, SimCLR and BYOL.

Category	**Component**	**Average Time (s)**		
		Cross-Entropy	**SimCLR**	**BYOL**
Normal Training	Data Loading	1.0135	1.3731	1.3717
	Forward Propagation	2.9785	5.5254	11.5720
	Backward Propagation	6.3665	14.2406	14.0334
	Model Update	0.6136	0.7443	0.7716
	Total Normal	**10.9721**	**21.8834**	**27.7487**
SGR Overhead	Gradient Replay	0.8367	0.8376	1.0204
	Gradient Recording	0.3664	0.4058	0.4720
	Total SGR	**1.2031**	**1.2434**	**1.4924**
MGR Overhead	Gradient Replay	0.7373	0.7220	0.8799
	Gradient Recording	0.3893	0.4190	0.4721
	Total MGR	**1.1266**	**1.1410**	**1.3520**

Our predefined replay interval is $K = 2$ (i.e., active and replay epochs alternate sequentially). Thus, following the E-th epoch, the training process can be viewed as an alternation between normal training and SGR/MGR-related operations, each accounting for half of the subsequent training epochs. Let T_{total} denote the total number of training epochs and E represent the replay start epoch. We define N as the time cost of a normal training epoch, and R as the overhead associated with SGR or MGR, given that their computational costs are approximately equivalent. Consequently, the speedup factor of the model can be quantified as follows:

$$\text{Speedup} = \frac{T_{\text{total}} \times N}{\frac{T_{\text{total}}-E}{2} \times (N + R) + E \times N}, \tag{11}$$

where the numerator and denominator represent the time overhead of not using Epoch Pruning and using Epoch Pruning, respectively. However, the overhead of the normal training N and the Epoch Pruning R are influenced by different factors. Disregarding the impact of hardware and software on model training speed, the computational time complexity for a normal training epoch $N = \mathcal{O}(D \cdot P \cdot A)$, where D, P, and A denote the dataset scale, parameter scale, and training algorithm complexity. Since we generate gradients batch by batch, and perform replay for every parameter, Epoch Pruning cost R is only determined by D and P, e.g., $R = \mathcal{O}(D \cdot P)$. However, the algorithmic complexity A varies significantly across different methods. For instance, as shown in Table 9, given the same dataset and model architecture, a self-supervised algorithm like SimCLR incurs substantially higher normal training time per epoch than standard cross-entropy-based supervised learning. Conversely, the time overhead for gradient recording and gradient replay remains largely consistent.

As indicated in the Table 9, the training overhead for SGR and MGR is approximately 0.1 times that of normal training (rounded to one decimal place). Consequently, when calculating the speedup rate of Epoch Pruning, we treat the computational cost of each replay epoch as equivalent to 0.1 active epochs. While this ratio varies across different methods, e.g., being significantly lower than 0.1 for algorithms like SimCLR and BYOL, we uniformly adopt the 0.1 ratio to compute the speedup, retaining one decimal place as a rough estimate. As is evident from the Table 9, the acceleration capability of our methods is substantial.

F Within-Epoch Gradient Gaussianity

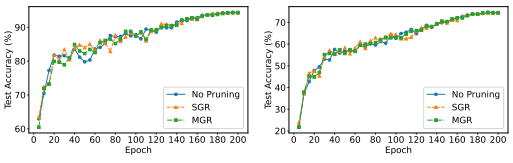
Here, we provide an empirical analysis to test whether the gradient distribution of a specific parameter within the same epoch follows a Gaussian distribution, providing support for the use of Gaussian distributions to generate replayed gradients in this work. In this study, the one-sample Kolmogorov-Smirnov (K-S) test (Massey Jr, 1951) was employed to assess the goodness-of-fit between the empirical distribution of the gradient and a theoretical Gaussian distribution. The K-S test evaluates this by calculating the maximum absolute difference between the empirical cumulative distribution function of the sample and the cumulative distribution function of the normal distribution. The null hypothesis (H_0) for this test assumes that, for the same parameter, its gradients within the epoch are drawn from a Gaussian distribution. We set the significance level (α) at 0.05. If the resulting p-value is less than 0.05, the null hypothesis is rejected, indicating a significant deviation from normality. We use the gradients produced by training ResNet (He et al., 2016) with an SGD-cosine configuration to examine this. We used the same experimental setup as in Table 1. Since the model has too many parameters, we select parameters with different functions for statistical analysis. We computed the proportion of weight-parameter gradients that follow a Gaussian distribution for the first convolutional layer (64×3×3×3; conv), the first fully connected layer (512×512; fc), and the first Batch Normalization layer (64; bn). The tests covered different training stages (numbers of epochs) on two datasets, CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). When calculating the proportions, we excluded parameters whose standard deviation is less than 1e-10. As shown in the Table 10, most of the parameters follow a Gaussian distribution, especially in the later training stage with replay enabled (after 100 epochs), which demonstrates the feasibility of the proposed methods for replaying gradients using a Gaussian distribution in the later stage of model training.

Table 10: The proportion (in percentage) of parameters whose within-epoch gradient distribution follows a Gaussian distribution.

Epoch	CIFAR-10			CIFAR-100		
	conv	fc	bn	conv	fc	bn
50	61.03	85.54	74.19	75.06	97.55	86.67
100	78.20	91.91	87.50	92.12	98.27	95.74
120	89.74	92.77	92.11	93.46	98.56	95.74
150	94.18	93.92	94.29	92.04	98.75	95.74
170	94.58	92.44	94.29	96.94	98.11	97.83

G Performance Comparison over Training Epochs

We conduct experiments to verify whether, during training, the proposed SGR and MGR can closely match the training progress of conventional training without Epoch Pruning. We perform experiments on CIFAR-10 and CIFAR-100 using the SGD-Cosine configuration. As shown in Figure 5, although SGR and MGR replay gradients and skip training during the Replay Epoch, the model's training progress is almost unaffected, with no significant performance fluctuations caused by gradient replay. This demonstrates that the replayed gradients of the proposed methods can effectively fulfill the functional role of the actual gradients.



(a) CIFAR-10 (b) CIFAR-100

Figure 5: Accuracy curves over training epochs for SGR, MGR, and the No Pruning baseline on CIFAR-10 and CIFAR-100.

H Proofs and Analysis of Theorem 1 and 2

In this section, we provide the detailed proofs for the descent bounds of standard SGD, SGR, and MGR. Let $\theta_{t+1} = \theta_t - \eta \hat{g}_t$ denote the parameter update rule, where $\hat{g}_t$ is the gradient estimator.

H.1 Proof of the Base Descent Lemma and Standard SGD Bound

Lemma 1. Under the assumption of L-smoothness, for any stochastic gradient estimator $\hat{g}_t$, the expected descent $\Delta = L(\theta_t) - \mathbb{E}[L(\theta_{t+1})]$ satisfies:

$$\Delta \geq \eta \langle \nabla L(\theta_t), \mathbb{E}[\hat{g}_t] \rangle - \frac{L\eta^2}{2} \left(\|\mathbb{E}[\hat{g}_t]\|^2 + \text{Tr}(\text{Var}(\hat{g}_t)) \right). \tag{12}$$

Proof of Lemma 1: Let the update step be $s_t = -\eta \hat{g}_t$. By the Fundamental Theorem of Calculus, the exact change in the loss function is:

$$\begin{aligned} L(\theta_{t+1}) - L(\theta_t) &= \int_0^1 \langle \nabla L(\theta_t + \tau s_t), s_t \rangle d\tau \\ &= \int_0^1 \langle \nabla L(\theta_t) + \nabla L(\theta_t + \tau s_t) - \nabla L(\theta_t), s_t \rangle d\tau \\ &= \int_0^1 \langle \nabla L(\theta_t), s_t \rangle d\tau + \int_0^1 \langle \nabla L(\theta_t + \tau s_t) - \nabla L(\theta_t), s_t \rangle d\tau. \end{aligned} \tag{13}$$

Since $\langle \nabla L(\theta_t), s_t \rangle$ is independent of τ, the first term simplifies as:

$$\int_0^1 \langle \nabla L(\theta_t), s_t \rangle d\tau = \langle \nabla L(\theta_t), s_t \rangle \int_0^1 1 d\tau = \langle \nabla L(\theta_t), s_t \rangle. \tag{14}$$

For the second term, we apply the Cauchy-Schwarz inequality ($\langle a, b \rangle \leq \|a\| \|b\|$) and the L-Lipschitz continuity assumption ($\|\nabla L(\theta_t + \tau s_t) - \nabla L(\theta_t)\| \leq L\|\tau s_t\| = L\tau \|s_t\|$):

$$\begin{aligned} \int_0^1 \langle \nabla L(\theta_t + \tau s_t) - \nabla L(\theta_t), s_t \rangle d\tau &\leq \int_0^1 \|\nabla L(\theta_t + \tau s_t) - \nabla L(\theta_t)\| \cdot \|s_t\| d\tau \\ &\leq \int_0^1 (L\tau \|s_t\|) \cdot \|s_t\| d\tau \\ &= L\|s_t\|^2 \int_0^1 \tau d\tau \\ &= L\|s_t\|^2 \left[\frac{\tau^2}{2} \right]_0^1 = \frac{L}{2} \|s_t\|^2. \end{aligned} \tag{15}$$

Combining Eq. (14) and Eq. (15), we obtain the quadratic upper bound:

$$L(\theta_{t+1}) - L(\theta_t) \leq \langle \nabla L(\theta_t), s_t \rangle + \frac{L}{2} \|s_t\|^2. \tag{16}$$

Substituting $s_t = -\eta \hat{g}_t$ back into Eq. (16):

$$\begin{aligned} L(\theta_{t+1}) &\leq L(\theta_t) + \langle \nabla L(\theta_t), -\eta \hat{g}_t \rangle + \frac{L}{2} \| - \eta \hat{g}_t\|^2 \\ &= L(\theta_t) - \eta \langle \nabla L(\theta_t), \hat{g}_t \rangle + \frac{L\eta^2}{2} \|\hat{g}_t\|^2. \end{aligned} \tag{17}$$

Taking the mathematical expectation $\mathbb{E}[\cdot]$ on both sides with respect to the stochastic gradient $\hat{g}_t$:

$$\mathbb{E}[L(\theta_{t+1})] \leq L(\theta_t) - \eta \langle \nabla L(\theta_t), \mathbb{E}[\hat{g}_t] \rangle + \frac{L\eta^2}{2} \mathbb{E}[\|\hat{g}_t\|^2]. \tag{18}$$

Next, we expand the expected squared norm $\mathbb{E}[\|\hat{g}_t\|^2]$. Let $X = \hat{g}_t$ and its mean $\mu_X = \mathbb{E}[X]$. The Euclidean norm can be rewritten as an inner product:

$$\begin{aligned} X^T X &= (\mu_X + (X - \mu_X))^T (\mu_X + (X - \mu_X)) \\ &= \mu_X^T \mu_X + \mu_X^T (X - \mu_X) + (X - \mu_X)^T \mu_X + (X - \mu_X)^T (X - \mu_X). \end{aligned} \tag{19}$$

Since $\mu_X^T (X - \mu_X)$ is a scalar, it equals its transpose $(X - \mu_X)^T \mu_X$. Also, note that for any column vector v, $v^T v = \text{Tr}(v^T v) = \text{Tr}(v v^T)$. Thus:

$$X^T X = \|\mu_X\|^2 + 2\mu_X^T (X - \mu_X) + \text{Tr}((X - \mu_X)(X - \mu_X)^T). \tag{20}$$

Taking the expectation $\mathbb{E}[\cdot]$ on both sides, and noting that $\mathbb{E}[X - \mu_X] = \mathbb{E}[X] - \mu_X = 0$:

$$\begin{aligned}\mathbb{E}[\|X\|^2] &= \|\mu_X\|^2 + 2\mu_X^T \mathbb{E}[X - \mu_X] + \text{Tr}(\mathbb{E}[(X - \mu_X)(X - \mu_X)^T]) \\ &= \|\mu_X\|^2 + 0 + \text{Tr}(\text{Var}(X)).\end{aligned} \tag{21}$$

Substituting this identity ($\mathbb{E}[\|\hat{g}_t\|^2] = \|\mathbb{E}[\hat{g}_t]\|^2 + \text{Tr}(\text{Var}(\hat{g}_t))$) into Eq. (18), we get:

$$\mathbb{E}[L(\theta_{t+1})] \leq L(\theta_t) - \eta \langle \nabla L(\theta_t), \mathbb{E}[\hat{g}_t] \rangle + \frac{L\eta^2}{2} \left(\|\mathbb{E}[\hat{g}_t]\|^2 + \text{Tr}(\text{Var}(\hat{g}_t)) \right). \tag{22}$$

Rearranging the terms to isolate $\Delta = L(\theta_t) - \mathbb{E}[L(\theta_{t+1})]$ yields Eq. (12).

Corollary (Standard SGD Bound). For standard SGD, the true mini-batch gradient g_t satisfies $\mathbb{E}[g_t] = \nabla L(\theta_t)$ and $\text{Tr}(\text{Var}(g_t)) = V_t$. Substituting these into Lemma 1:

$$\begin{aligned}\Delta_{sgd} &\geq \eta \langle \nabla L(\theta_t), \nabla L(\theta_t) \rangle - \frac{L\eta^2}{2} \left(\|\nabla L(\theta_t)\|^2 + V_t \right) \\ &= \eta \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2}{2} \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2}{2} V_t \\ &= \eta \left(1 - \frac{L\eta}{2} \right) \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2}{2} V_t.\end{aligned} \tag{23}$$

As shown in Eq. (23), for standard SGD, the expected descent lower bound is directly penalized by the local gradient variance trace V_t at step t. Similarly, if one naively replays uncalibrated historical gradients at the onset of a replay epoch when the parameter trajectory has not yet stabilized, a large historical variance V_{t-1} combined with gradient drift can dominate the first-order descent term, pushing the expected descent bound negative and causing training fluctuations (as empirically observed in Figure 2).

H.2 Proof of Theorem 1 (SGR Bound)

In SGR, the gradient is generated by $\hat{g}_t = \tilde{g}_{sgr} = w_{sgr} \cdot (\mu + z \cdot \sigma)$, where $z \sim \mathcal{N}(0, I)$ so that $\mathbb{E}[z] = \mathbf{0}$ and $\mathbb{E}[zz^T] = \text{Var}(z) = I$. Under the assumption, $\mu = \nabla L(\theta_t) + \xi$ (where $||\xi|| \leq \epsilon$) and $V_{t-1} = \sum \sigma_i^2$. We can rewrite the generation function as matrix multiplication:

$$\tilde{g}_{sgr} = w_{sgr}\mu + w_{sgr}\text{diag}(\sigma)z. \tag{24}$$

The expected value (first moment) is:

$$\begin{aligned}\mathbb{E}[\tilde{g}_{sgr}] &= \mathbb{E}[w_{sgr}\mu + w_{sgr}\text{diag}(\sigma)z] \\ &= w_{sgr}\mu + w_{sgr}\text{diag}(\sigma)\mathbb{E}[z] \\ &= w_{sgr}(\nabla L(\theta_t) + \xi) = w_{sgr}\nabla L(\theta_t) + w_{sgr}\xi.\end{aligned} \tag{25}$$

To calculate the variance matrix from its definition $\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^T]$, we first find the centered variable:

$$\tilde{g}_{sgr} - \mathbb{E}[\tilde{g}_{sgr}] = (w_{sgr}\mu + w_{sgr}\text{diag}(\sigma)z) - w_{sgr}\mu = w_{sgr}\text{diag}(\sigma)z. \tag{26}$$

Then the variance matrix is:

$$\begin{aligned}\text{Var}(\tilde{g}_{sgr}) &= \mathbb{E}[(w_{sgr}\text{diag}(\sigma)z)(w_{sgr}\text{diag}(\sigma)z)^T] \\ &= \mathbb{E}[w_{sgr}\text{diag}(\sigma)zz^T\text{diag}(\sigma)^T w_{sgr}] \\ &= w_{sgr}^2\text{diag}(\sigma)\mathbb{E}[zz^T]\text{diag}(\sigma) \\ &= w_{sgr}^2\text{diag}(\sigma)I\text{diag}(\sigma) = w_{sgr}^2\text{diag}(\sigma^2).\end{aligned} \tag{27}$$

Taking the trace yields:

$$\text{Tr}(\text{Var}(\tilde{g}_{sgr})) = \text{Tr}(w_{sgr}^2\text{diag}(\sigma^2)) = w_{sgr}^2 \sum_i \sigma_i^2 = w_{sgr}^2 V_{t-1}. \tag{28}$$

Now, substituting $\mathbb{E}[\tilde{g}_{sgr}] = w_{sgr}\nabla L(\theta_t) + w_{sgr}\xi$ and $Tr(Var(\tilde{g}_{sgr})) = w_{sgr}^2 V_{t-1}$ into Lemma 1 yields drift-dependent terms $\eta w_{sgr}(1 - L\eta w_{sgr})\langle \nabla L(\theta_t), \xi \rangle - \frac{L\eta^2 w_{sgr}^2}{2}\|\xi\|^2$. By applying the

Cauchy-Schwarz inequality ($|\langle \nabla L(\theta_t), \xi \rangle| \leq \|\nabla L(\theta_t)\| \|\xi\| \leq G\epsilon$) and $\|\xi\|^2 \leq \epsilon^2$, we absorb these bounded drift terms into the residual $\mathcal{O}(\epsilon)$:

$$\Delta_{sgr} \geq \eta \langle \nabla L(\theta_t), w_{sgr} \nabla L(\theta_t) \rangle - \frac{L\eta^2}{2} \left(\|w_{sgr} \nabla L(\theta_t)\|^2 + w_{sgr}^2 V_{t-1} \right) - \mathcal{O}(\epsilon). \tag{29}$$

We expand the inner product $\langle \nabla L(\theta_t), w_{sgr} \nabla L(\theta_t) \rangle = w_{sgr} \langle \nabla L(\theta_t), \nabla L(\theta_t) \rangle = w_{sgr} \|\nabla L(\theta_t)\|^2$, and expand the norm squared $\|w_{sgr} \nabla L(\theta_t)\|^2 = w_{sgr}^2 \|\nabla L(\theta_t)\|^2$:

$$\begin{aligned} \Delta_{sgr} &\geq \eta \left(w_{sgr} \|\nabla L(\theta_t)\|^2 \right) - \frac{L\eta^2}{2} \left(w_{sgr}^2 \|\nabla L(\theta_t)\|^2 + w_{sgr}^2 V_{t-1} \right) - \mathcal{O}(\epsilon) \\ &= \eta w_{sgr} \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2 w_{sgr}^2}{2} \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2 w_{sgr}^2}{2} V_{t-1} - \mathcal{O}(\epsilon). \end{aligned} \tag{30}$$

Factoring out the common term $\|\nabla L(\theta_t)\|^2$ from the first two terms:

$$\Delta_{sgr} \geq \eta w_{sgr} \left(1 - \frac{L\eta w_{sgr}}{2} \right) \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2 w_{sgr}^2}{2} V_{t-1} - \mathcal{O}(\epsilon). \tag{31}$$

This completes the proof of Theorem 1.

Remark on Theorem 1: This theorem explains how Progressive Linear Weighting (PLW) stabilizes the transition from active training to gradient replay (comparing SGR with $w_{sgr} \in (0, 1)$ against naive full-step replay with $w_{sgr} = 1$). In Eq. (9), the first-order descent gain scales linearly with w_{sgr}, whereas the second-order variance penalty $-\frac{L\eta^2 w_{sgr}^2}{2} V_{t-1}$ scales quadratically with w_{sgr}^2. At the onset of replay ($T_{current}$ close to E), where historical gradient statistics are most susceptible to staleness and high variance V_{t-1}, having $w_{sgr}^2 \ll w_{sgr}$ acts as a quadratic variance damper. This prevents the historical variance penalty from dominating the linear descent term, ensuring Δ_{sgr} remains stable.

H.3 Proof of Theorem 2 (MGR Bound)

In MGR, the synthetic gradient utilizes the Exponential Moving Average (EMA) of historical statistics: $\hat{g}_t = \tilde{g}_{mgr} = \bar{\mu} + z \cdot \bar{\sigma} = \bar{\mu} + \text{diag}(\bar{\sigma}) z$. Under the assumption, the EMA provides an approximately unbiased mean $\overline{\mu} = \nabla L(\theta_t) + \xi$, where ξ bounds the accumulated gradient drift such that $||\xi|| \leq \epsilon$. Let the trace of the smoothed variance be $\bar{V}_t = \sum \bar{\sigma}_i^2$.

We evaluate the expected value (first moment):

$$\begin{aligned} \mathbb{E}[\tilde{g}_{mgr}] &= \mathbb{E}[\bar{\mu} + \text{diag}(\bar{\sigma}) z] \\ &= \overline{\mu} + diag(\overline{\sigma}) \mathbb{E}[z] = \nabla L(\theta_t) + \xi. \end{aligned} \tag{32}$$

For the variance matrix, the centered variable is $\tilde{g}_{mgr} - \mathbb{E}[\tilde{g}_{mgr}] = \text{diag}(\bar{\sigma}) z$. Thus:

$$\begin{aligned} \text{Var}(\tilde{g}_{mgr}) &= \mathbb{E}[(\text{diag}(\bar{\sigma}) z)(\text{diag}(\bar{\sigma}) z)^T] \\ &= \mathbb{E}[\text{diag}(\bar{\sigma}) z z^T \text{diag}(\bar{\sigma})^T] \\ &= \text{diag}(\bar{\sigma}) \mathbb{E}[z z^T] \text{diag}(\bar{\sigma}). \\ &= \text{diag}(\bar{\sigma}) I \text{diag}(\bar{\sigma}) = \text{diag}(\bar{\sigma}^2). \end{aligned} \tag{33}$$

Taking the trace yields:

$$\text{Tr}(\text{Var}(\tilde{g}_{mgr})) = \text{Tr}(\text{diag}(\bar{\sigma}^2)) = \sum_i \bar{\sigma}_i^2 = \bar{V}_t. \tag{34}$$

Substituting $\mathbb{E}[\tilde{g}_{mgr}] = \nabla L(\theta_t) + \xi$ and $Tr(Var(\tilde{g}_{mgr})) = \overline{V}_t$ into Lemma 1, and similarly bounding the drift terms $\eta(1 - L\eta)\langle \nabla L(\theta_t), \xi \rangle - \frac{L\eta^2}{2} \|\xi\|^2$ via the Cauchy-Schwarz inequality into $\mathcal{O}(\epsilon)$:

$$\Delta_{mgr} \geq \eta \langle \nabla L(\theta_t), \nabla L(\theta_t) \rangle - \frac{L\eta^2}{2} \left(\|\nabla L(\theta_t)\|^2 + \bar{V}_t \right) - \mathcal{O}(\epsilon). \tag{35}$$

Applying the inner product identity $\langle \nabla L(\theta_t), \nabla L(\theta_t) \rangle = \|\nabla L(\theta_t)\|^2$ and expanding the terms:

$$\Delta_{mgr} \geq \eta \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2}{2} \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2}{2} \bar{V}_t - \mathcal{O}(\epsilon). \tag{36}$$

Factoring out the common term $\|\nabla L(\theta_t)\|^2$ completes the proof:

$$\Delta_{mgr} \geq \eta \left(1 - \frac{L\eta}{2}\right) \|\nabla L(\theta_t)\|^2 - \frac{L\eta^2}{2} \bar{V}_t - \mathcal{O}(\epsilon). \tag{37}$$

This completes the proof of Theorem 2.

Remark on Theorem 2: This theorem demonstrates that MGR acts as a temporally smoothed gradient replay mechanism. In single-epoch replay without momentum, the variance trace V_{t-1} relies solely on a single preceding epoch and can fluctuate across epochs. Conversely, the EMA mechanism in MGR aggregates historical moments across multiple active epochs, acting as a temporal low-pass filter that satisfies $\bar{V}_t \leq \max_{k \leq t}(V_k)$. Consequently, whenever a single preceding epoch exhibits abnormally high variance ($V_{t-1} > \bar{V}_t$), MGR suppresses the outlier variance penalty and provides a more stable optimization trajectory.