arXiv:2505.24452v4 [cs.LG] 8 Dec 2025

Stepsize anything: A unified learning rate schedule for budgeted-iteration training

Anda Tang[1] **Yiming Dong**[1]* **Yutao Zeng**[2] **Xun Zhou**[2] **Zhouchen Lin**[1,3,4]†
[1]State Key Lab of General AI, School of Intelligence Science and Technology, Peking University
[2]ByteDance Seed
[3]Institute for Artificial Intelligence, Peking University
[4]Pazhou Laboratory (Huangpu), Guangzhou, Guangdong, China
tanganda@pku.edu.cn, yimingdong_ml@outlook.com,
yutao.zeng@outlook.com, zhouxun@bytedance.com, zlin@pku.edu.cn

Abstract

The expanding computational costs and limited resources underscore the critical need for budgeted-iteration training, which aims to achieve optimal learning within predetermined iteration budgets. While learning rate schedules fundamentally govern the performance of different networks and tasks, particularly in budgeted-iteration scenarios, their design remains largely heuristic, lacking theoretical foundations. In addition, the optimal learning rate schedule requires extensive trial-and-error selection, making the training process inefficient. In this work, we propose the Unified Budget-Aware (UBA) schedule, a theoretically grounded learning rate schedule that consistently outperforms commonly-used schedules among diverse architectures and tasks under different constrained training budgets. First, we bridge the gap by constructing a novel training budget-aware optimization framework, which explicitly accounts for the robustness to landscape curvature variations. From this framework, we derive the UBA schedule, controlled by a single hyper-parameter φ that provides a trade-off between flexibility and simplicity, eliminating the need for per-network numerical optimization. Moreover, we establish a theoretical connection between φ and the condition number, adding interpretation and justification to our approach. Besides, we prove the convergence for different values of φ. We offer practical guidelines for its selection via theoretical analysis and empirical results. Extensive experimental results show that UBA *consistently surpasses* the commonly-used schedules across diverse vision and language tasks, spanning network architectures (e.g., ResNet, OLMo) and scales, under different training-iteration budgets.

1 Introduction

Deep learning has achieved remarkable success across a wide range of domains, including computer vision and natural language processing. However, despite continual advancements in hardware technologies [45, 53], the training cost of neural networks has increased dramatically due to the growing scale of models and datasets [4, 8, 47, 48]. As a result, resource constraints, including computational power, memory, energy consumption, and time budgets, are emerging as significant bottlenecks in the training process [41, 58]. These challenges highlight the pressing need for budgeted training, which aims to achieve optimal model performance under fixed hardware and limited time.

*Work was done during an internship at ByteDance Seed.
†Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

While existing budgeted training studies broadly address resource efficiency, a critical yet under-explored direction within budgeted training is achieving the best possible model performance under strictly fixed iteration constraints. This scenario is common and practically significant where practitioners work under limited computational or time budgets [30], and in extreme cases, models have to be completed within a few training iterations due to resource exhaustion. To formalize this specific research problem, we introduce the term 'budgeted-iteration training' , distinguishing it from the broader scope of budgeted training.

Budgeted-iteration training has received growing attention in research, given its significant real-world applicability. Several studies have developed relevant techniques that align with its goals. Smith et al. [43] propose the cyclical learning rate schedule (CLR), improving accuracy in fewer iterations without tuning [44]. Li et al. [30] introduce budget-aware adaptations for existing learning rate schedules. Chen et al. [5] propose a novel learning rate schedule called Reflected Exponential (REX). These approaches are primarily based on learning rate designing. Learning rate scheduling highlights key advantages: (i) It plays a critical role in general training of diverse neural architectures across tasks. (ii) It has demonstrated suitable and competitive in fixed training iteration budgets [20]. (iii) It is plug-and-play, requiring minimal adjustments to the underlying model architecture, which makes them easily adaptable to various deep learning frameworks. Leveraging these advantages, we adopt the learning rate design approach for budgeted-iteration training.

Despite their advantages, most learning rate schedules, whether tailored for budgeted-iteration training or standard training performance, are still heuristic and lack rigorous theoretical grounding. In addition, existing schedules typically rely on manually designed rules or empirical tuning. Consequently, selecting an optimal schedule often involves extensive trial-and-error, incurring substantial cost in terms of time and computation [34].

A natural question arises: Does there exist a theoretically grounded, unified schedule that eliminates heuristic selection while maintaining robust performance across tasks, networks, scales and training budgets?

In this paper, we provide an affirmative answer by proposing a theoretically grounded schedule. The proposed learning rate schedule should consistently outperforms existing schedules among diverse architectures and tasks under different constrained training budgets. By doing so, it avoids choosing suitable learning rate schedule after multiple trials for network training.

To achieve this, we first bridge the gap by constructing a unified budget-aware training optimization framework, which incorporates the robustness to landscape curvature variations induced by data distribution, sampling, network architectures and optimization. The framework employs a min-max optimization approach to guarantee performance under worst-case conditions, since the minimization of learning rate is operated on an upper bound. Then we obtain numerical solutions by gradient projection methods. To eliminate the need for repeated numerical optimization when applying our method to different networks, we propose a universal parametric function that approximates numerical solutions. We nominate the resulting schedule Unified Budget-Aware (UBA) schedule. It requires tuning only a single hyper-parameter φ, reducing the overhead of per-network numerical optimization. Moreover, we establish a theoretical connection between φ and the condition number, adding interpretation and optimization difficulty-aware theoretical grounding. Besides, we prove the convergence for different values of φ. These theoretical analysis along with empirical results offer practical guidelines for φ selection. We evaluate UBA through comprehensive experiments across vision and language tasks, spanning diverse architectures and iteration budgets. Specifically, for vision tasks, the UBA schedule demonstrates *consistent superiority* over baselines across all evaluated datasets and model scales under varying training iterations. For language tasks, we validate the effectiveness of UBA through extensive benchmarks with OLMo model(36M, 73M, 150M and 300M parameters). Results show that UBA achieves state-of-the-art performance across on approximately half of the benchmarks, and consistently outperforms baselines on the average scores. Moreover, our performance improvements come with negligible computational overhead, which constitutes one of its key advantages.

Main contributions:

1. We construct a unified budget-aware training optimization framework that inherently adapts to landscape curvature variations, enhancing training robustness of leaning rates.

2. We propose the Unified Budget-Aware (UBA) schedule from our constructed optimization problem, controlled by a single hyper-parameter φ that provides a trade-off between flexibility and simplicity, i.e. adaptive curvature adjustment and minimal tuning cost.
3. We prove the convergence under φ and derive practical guidelines for its selection through analysis and experiments. Besides, theoretical analysis and empirical results show that φ is related to optimization difficulty such as condition number.
4. We perform experiments and demonstrate that UBA surpasses the commonly-used schedules across diverse vision and language tasks, spanning network architectures (e.g., ResNet, OLMo) and scales, under different training-iteration budgets. Moreover, our performance improvements come with negligible computational overhead, which constitutes one of its key advantages.

2 Related work

Budgeted training Researchers face significant challenges in achieving optimal model performance under fixed hardware and limited time. To address these challenges, the concept of budgeted training has gained increasing attention, exploring techniques including computation efficiency, model compression, training stability and convergence improvement [41]. It focuses on: (i) emphasizing the allocation of resources, such as the balance between the model size and the amount of data [2, 6, 24, 28]. (ii) finding optimal configurations or improving performance within the given compute or time budget, such as memory efficiency and computation reduction [17, 26, 31, 38], optimization learning rate schedules [5, 30, 43, 44], batch size [17] and other weight averaging method [26, 27].

Within the context of budgeted training, a key area that remains under-explored is achieving the best possible model performance within a fixed number of iterations, i.e. 'budgeted-iteration training'. In this domain, learning rate scheduling based methods are particularly aligned with the objectives of budgeted-iteration training. Smith et al. [43] propose a new learning rate schedule, named cyclical learning rates (CLR). It improves accuracy in fewer iterations without tuning and is relevant to super-convergence phenomenon [44]. Li et al. [30] introduce an alternative setting of existing learning rate schedules for budgeted training. Chen et al. [5] propose the reflected exponential schedule (REX) via a profile and sampling fashion. Zhang et al. [57] introduce the finite horizon optimization framework and apply it to linear programming, which optimizes the algorithm performance under a strict iteration budget. Learning rate based approaches achieve robust and high-performing results under various lengths of training iterations, which corresponds to our purpose in this work, i.e. budgeted-iteration training [34]. Besides, this approach is plug-and-play, requiring no substantial alterations to the underlying model structure, making it readily adaptable to various deep network frameworks. Therefore, we explore the proper learning rate schedule to achieve budgeted-iteration training.

Learning rate schedule The learning rate plays a pivotal role in controlling the optimization process during network training. The common scheme is the step decay schedule. A typical instance decreases the learning rate by a decaying scalar 0.1 after 50% epochs and by a decaying scalar 0.01 after 75% epochs [21]. Then Loshchilov et al. [33] observe that sharp decreases may prevent models from escaping local minima and propose the cosine schedule function, which is the most popular schedule for language model pretraining.

Pan et al. [36] propose Eigencurve, and demonstrate that minimax-optimal convergence can be achieved for quadratic objectives when the Hessian's eigenvalue distribution exhibits substantial skewness. Defazio et al. [11] developed the theoretically-grounded framework for learning rate scheduling, establishing optimal linear decay schedules and refinements through rigorous mathematical derivation. Their work additionally provides the most extensive empirical evaluation of scheduling approaches to date. Although some schedules include the CLR [43], REX [5], Warmup-Stable-Decay (WSD) [25] and schedule from multi-power law [34], there is no consensus on the optimal choice. In addition, the detail can be found in Appendix F, including some works focus on adaptive learning rate methods.

Learning rate design is not only significant in general training, but also critical in budgeted-iteration training which still remains a topic of debate. Some analyses advocate for small, constant learning rates to ensure stability and convergence [12]. On the contrast, one prevailing hypothesis suggests that

large learning rates may facilitate crossing over sharp local minima in the optimization landscape [56]. Despite the lack of comprehensive theoretical explanations, a range of learning rate schedules inspired by the above analyses as heuristic guidelines has been widely adopted in practice, using variable learning rates to budgeted-iteration training [5, 30]. In this work, we explore learning rate schedule from optimization problem tailored to budgeted-iteration training, aiming to balance iteration budget constraints and generalization.

3 Budgeted-iteration training

3.1 Finite optimization under limited training iterations

To design a one-size-fits-all learning rate schedule, we construct a robust optimization model of learning rates across varying training conditions (see more conceptual illustration in Appendix A). Specifically, we aim to guarantee minimal loss within constrained training iterations under the worst-case conditions, proposing a budget-iteration-aware framework for learning rate optimization.

Definition (Finite optimization): Let $f(W, D)$ denote the function parameterized by a given neural network with parameters $W \in \mathbb{R}^N$ on the dataset D. Let ξ denote the data sampling on the dataset D, and let $\mathcal{F}$ be a function class.Let $\mathcal{L}$ be the loss function. Let η_t be the learning rate at the t-th iteration, T be a maximum number of training iterations, and t be the current learning step. A finite optimization is

$$\begin{aligned} \min_{\eta_1,\eta_2,\ldots,\eta_{T-1}} \max_{f\in\mathcal{F}} \quad & \mathcal{L}(f(W_T,\xi)) \\ s.t. \quad & W_{t+1} = W_t - \eta_t \nabla \mathcal{L}(f(W_t,\xi)) \\ & t = 0, 1, 2, ..., T-1 \end{aligned} \tag{1}$$

In the optimization model 1, the constraint represents the stochastic gradient descent process. The maximizing of $\mathcal{L}(f(W_T, \xi))$ represents the the worst-case among the training process on the network f. Then it minimizes the worst-case loss within given iterations, embodying its budget-aware property. By formulating the problem as a min-max optimization, we identify learning rates $\eta_t (t = 1, 2, \cdots, T-1)$ that are resilient to the uncertainties introduced by different training configuration, uniformly throughout the optimization trajectory.

The challenge lies in characterizing the f within the optimization process. f is primarily determined by variations in parameter configuration, datasets characteristics, batch ordering and network architecture. From an optimization standpoint, we assume that the characteristics of f shaped by these factors can be captured by the loss landscape of f. Therefore, we can analyze f by approximating its loss using a quadratic expansion around nearby strict local optima. Specifically, during the optimization process, the loss surface in the vicinity of the optimization trajectory can be approximated by sequence of strict local optima (or at least by one optimum), denoted as $\bar{W}^{(k)} \quad (k = 1, 2, \ldots K) \quad K \in \mathbb{Z}^+$, with the final optimum represented as $\bar{W}^K$. This approach enables us to capture the key features of the loss surface near these points. Therefore, the trajectory of optimization is impacted by the characteristics of the nearby strict local optima, since the optimization process is inherently shaped by the loss landscape and the key features of the surface are captured by these optima. Consequently, we can derive the learning rate within the optimization model by the information of these nearby strict local optima.

According to the second-order necessary condition for the strict minimum $\bar{W}^{(k)}$, this approximation is achieved through the positive semi-definite Hessian matrix $H_f^{(k)}(\xi) \in \mathbb{R}^{N\times N}$. In addition, the optimization problem (1) will be programmed sequentially. Then objective function of the optimization problem (1) can be reformulated as $\mathcal{L}(f(\bar{W}^{(k)}, \xi)) + \frac{1}{2}(W_{T_{k+1}} - \bar{W}^{(k)})^\top H_f^{(k)}(\xi)(W_{T_{k+1}} - \bar{W}^{(k)}) \quad (k = 1, 2, \ldots K)$. Given that the networks under consideration possess sufficient capacity to fit the data, the loss at the optimal point for different f can be made small enough. Thus the term $\mathcal{L}(f(\bar{W}^{(k)}, \xi))$ can be reasonably neglected in our analysis. We obtain the following sequential optimization problem.

$$\min_{\eta_{1+T_k},\eta_{2+T_k},\ldots,\eta_{T_{k+1}}} \max_{f(\xi)} \quad \frac{1}{2}(W_{T_{k+1}} - \bar{W}^{(k)})^\top H_f^{(k)}(\xi)(W_{T_{k+1}} - \bar{W}^{(k)})$$
$$s.t. \quad W_{t+1} = W_t - \eta_t H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)}) \tag{2}$$
$$t = T_k + 1, T_k + 2, ..., T_{k+1} \quad (k = 1, 2, \ldots K-1)$$

where $T_1 = 0$ and $T_K = T - 1$.

By the first constraint $W_{t+1} = W_t - \eta_t H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)})$, we can obtain (see Appendix A for derivation)

$$\left\|W_{T_{k+1}} - \bar{W}^{(k)}\right\|_2^2 \leq \max_{\lambda_l^{(k)} \leq \lambda_i^{(k)} \leq \lambda_u^{(k)}} \left[\prod_{t=1+T_k}^{T_{k+1}} (1 - \eta_t \lambda_i^{(k)})\right]^2 \left\|W_{T_k} - \bar{W}^{(k)}\right\|_2^2 \tag{3}$$

where $\lambda_i^{(k)} \quad (i = 1, 2, \cdots, N)$ are the eigenvalues of $H_f^{(k)}(\xi)$ around the k-th optimum, which satisfy $0 < \lambda_l^{(k)} \leq \lambda_i^{(k)}(f(\xi)) \leq \lambda_u^{(k)}$ for all i, $u_i^{(k)} \quad (i = 1, 2, \cdots, N)$ denote N linearly independent eigenvectors and $s_i^{(k)}$ are the coefficients corresponding to the eigenvector components. Since the term $\left\|W_{T_k} - \bar{W}^{(k)}\right\|_2^2$ is a certain constant, the optimization process of weights $W_{T_{k+1}}$ is equivalent to the least upper bound of the $\prod_{t=1+T_k}^{T_{k+1}} (1 - \eta_t \lambda_i^{(k)})$. Thus, the optimization of learning rate schedule can be formulated as follow,

$$\min_{\eta_{1+T_k},\eta_{2+T_k},\ldots,\eta_{T_{k+1}}} \max_{\lambda_l^{(k)} \leq \lambda_i^{(k)} \leq \lambda_u^{(k)}} \prod_{t=1+T_k}^{T_{k+1}} \left[(1 - \eta_t \lambda_i^{(k)})\right]^2$$
$$s.t. \quad \eta_{1+T_k}, \eta_{2+T_k}, ..., \eta_{T_{k+1}} \in [\eta_{\min}, \eta_{\max}] \tag{4}$$
$$k = 1, 2, \ldots K$$

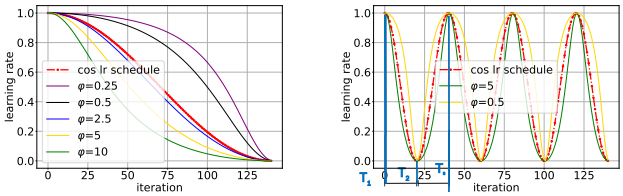
To solve the constrained min-max problem (4), we adopt an iterative projected gradient method that alternates between minimizing over the variables η_t and maximizing over the parameters $\lambda_i^{(k)}$. To avoid repeated numerical optimization, we fit the solutions with a parametric function. Details regarding the numerical solution and curve fitting process are provided in Appendix B. Then, we obtain the η_t within the interval $t \in [1 + T_k, T_{k+1}]$ as follows

$$\eta_t = (\eta_{\max} - \eta_{\min}) \frac{2(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)} + (k-1)\pi))}{2\varphi + (2-\varphi)(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)} + (k-1)\pi)} + \eta_{\min}, \tag{5}$$

where φ is the hyper-parameter controlling the variation speed of the learning rate η_t.

When setting $(K > 2)$, UBA extends to a multi-phase formulation, which offers three potential advantages. (i) Hierarchical optimization: by partitioning the budget-aware optimization into multiple phases, each near a different local minima, it improves approximation accuracy and captures the dynamic features of the loss surface. (ii) It helps escape saddle points, (iii) It generalizes the single-phase method, allowing for future extensions. Notably, a single-phase approach $(K = 2)$ remains effective, as the robust budget-aware model inherently derives the optimal schedule function, as shown in Figure 1(a). For consistency with common practice and to better isolate scheduling effects, this paper focuses primarily on the single-phase implementation. The multi-phase approach is also compared in Appendix E.9, with its schedule shown in Figure 1(b).

We name the proposed schedule Unified Budget-Aware (UBA) schedule for the following reasons. The robust budget-aware model minimizes the loss function **uniformly** across the optimization trajectory, resulting in stable performance. UBA provides a reliable, **unified** choice for practitioners, eliminating the need for case-by-case baseline comparisons and delivering consistent superiority across datasets, architectures, and training budgets. Lastly, UBA can **uniformly** approximate the behavior of existing schedules through simple parameter adjustments.



(a) (b)

Figure 1: Evolution of the learning rate in UBA schedule across training iterations.

3.2 Theoretical analysis

Proposition 1. *The fit function* (5) *is the exact closed-form solution to the min-max optimization problem:*

$$\min_{\eta_{1+T_k},\eta_{2+T_k},\ldots,\eta_{T_{k+1}}} \max_{\lambda_l^{(k)}\le\lambda_i^{(k)}\le\lambda_u^{(k)}} \prod_{t=1+T_k}^{T_{k+1}} \left[\left(1-\left(\left(\frac{1}{\lambda_l^{(k)}}-\frac{1}{\lambda_u^{(k)}}\right)\eta_t+\frac{1}{\lambda_u^{(k)}}\right)\lambda_i^{(k)}\right)\right]^2 \tag{6}$$

when the hyper-parameter φ are determined by $\lambda_l^{(k)}$ and $\lambda_u^{(k)}$ through the relation $\varphi = 2\frac{\lambda_u^{(k)}}{\lambda_l^{(k)}}$ and $\eta_{\max} = 1, \quad \eta_{\min} = 0.$

The min-max model in Proposition 1 represents a special case of our generalized optimization framework. We show that UBA is the exact solution to this special case optimization problem when the learning rate is scaled by $\left(\left(\frac{1}{\lambda_l^{(k)}}-\frac{1}{\lambda_u^{(k)}}\right)\eta_t+\frac{1}{\lambda_u^{(k)}}\right)$. It provides a theoretical foundation for our choice of learning rate instead of choosing it heuristically or empirically, adding rigor and justification to our approach. Moreover, in this case, φ is linked condition number. It indicates how the learning rate is shaped by the local curvature of the model, which could guide the optimization process more effectively. The relationship between φ and condition number suggests an adaptive nature for the learning rate. In regions with sharp curvatures (large condition number), φ reduces the learning rate more rapidly to avoid overshooting. Conversely, in flatter regions, φ allows the learning rate to remain large for several iterations, facilitating faster convergence. The transformation of the learning rate trend is shown in Figure 1. *This is a step toward establishing a principled connection between learning rate and local loss landscape geometry along with optimization difficulty.* By this special case, we generalize that φ is related to optimization difficulty. The empirical results support these conclusions, with further details in Section 4.3 and Appendix E.6.

Proposition 2. *Consider the training process within the interval $t \in [1+T_k, T_{k+1}]$. When the hyper-parameter φ is set sufficiently close to* 2*, the proposed learning rate scheduling formula reduces to the cosine learning rate schedule.*

By Proposition 1, the hyper-parameter φ is related to the optimization difficulty. In that case, the optimization difficulty is explicitly quantified as the condition number and $\varphi = 2\kappa$, where $\kappa = \frac{\lambda_u^{(k)}}{\lambda_l^{(k)}}$. Furthermore, Proposition 2 shows that when φ approaches 2, the proposed learning rate schedule converges to the standard cosine schedule. It means that, in regions where the optimization difficulty is manageable (i.e., the curvature is relatively flat), setting $\varphi \approx 2$ allows the learning rate schedule to naturally reduce to the cosine form. Besides, our schedule can approximate the behavior of existing schedules (e.g.step decay, cosine annealing, cyclic schedule or Rex schedule) through simple parameter adjustments, shown in Table 1. The detail can be found in Appendix C. More importantly, it outperforms these schedules by training convergence and final accuracy, supporting results can be found in experiment sections 4.1,4.2 and Appendix E.5.

Theorem 1. *Let $n_t = H_f^{(k)}(W_t - \bar{W}^{(k)}) - H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)})$ be the stochastic curvature noise introduced by sampling at iteration t. Assume that the sampling Hessian satisfies: $\mathbb{E}_\xi\left[n_t n_t^\top\right] \preceq \sigma^2 H_f^{(k)}$ for some constant σ. Denote $\tau := \frac{4\lambda_l^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(\varphi-2)\pi}$. If we*

Table 1: The adaptive simulation of existing schedules.

Schedule	Parameter adjustments	Schedule	Parameter adjustments
Cosine	$\varphi = 2$	Step	$\varphi = 0, \eta_{\max} = 0.5^k, T_{k+1} - T_k =$decaying step,$k = 1, 2, ...$
Exponential	$\varphi = 30$	Cyclic	$\varphi = 2, k \leftarrow k+1, T_{k+1} - T_k =$cyclic step ,$k = 1, 2, ...$
Rex	$\varphi = 0.8$	OneCycle	$\varphi = 2, k \leftarrow k+1, T_2 - T_1 =$ pct_start step

set the learning rate as the proposed form (5)*, the loss uncertainty introduced by stochastic gradient method within the interval* $t \in [1 + T_k, T_{k+1}]$ *can be bounded by two terms,*

For $\varphi > 2$***:***

$$\begin{aligned}
&\mathbb{E}\Big[\mathcal{L}(f(W_t,\xi)) - \mathcal{L}(f(\bar{W}^{(k)},\xi))\Big] \\
\leq&\Big(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\Big)^{\tau}\cdot\exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\lambda_u^{(k)}\|W_{1+T_k}-\bar{W}^{(k)}\|_2^2 \\
&+\sigma^2\sum_{i=_{1+T_k}}^{t}\eta_i^2\sum_{j=1}^{N}(\lambda_j^{(k)})^2\exp\Big(-2\lambda_j^{(k)}\eta_{\min}(t-i)\Big)\cdot\left(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi(i-T_k)}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\right)^{\tau}
\end{aligned} \tag{7}$$

For $\varphi < 2$***:***

$$\begin{aligned}
&\mathbb{E}\Big[\mathcal{L}(f(W_t,\xi)) - \mathcal{L}(f(\bar{W}^{(k)},\xi))\Big] \\
\leq&\Big(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\Big)^{-\tau}\cdot\exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\lambda_u^{(k)}\|W_{1+T_k}-\bar{W}^{(k)}\|_2^2 \\
&+\sigma^2\sum_{i=_{1+T_k}}^{t}\eta_i^2\sum_{j=1}^{N}(\lambda_j^{(k)})^2\exp\Big(-2\lambda_j^{(k)}\eta_{\min}(t-i)\Big)\cdot\left(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(i-T_k-0.5)}\right)^{-\tau}
\end{aligned} \tag{8}$$

4 Experiment results

We conduct comprehensive evaluations along three dimensions: (i) Modality diversity, including vision and language tasks; (ii) Training budget, including model scales (small to large) and training iterations (short to long); (iii) Ablation studies, such as parameter sensitivity and cross-optimizer performance. This evaluation strategy ensures the robustness and generalization of our findings.

Our implementation strictly follows the PyTorch official API, i.e. it is inherited from the base class `torch.optim.lr_scheduler.LRScheduler` (Appendix E.3). It maintains full compatibility with all `torch.optim` optimizers. The complete production-ready code is publicly available (see Appendix E.3 for link).

A key advantage of UBA lies in its critical parameter φ. While an optimal φ can further improve model performance, we intentionally fix its value across all tasks and architectures to ensure fair evaluation. We fix φ for SGD and AdamW respectively (see learning rate variations in Figure 1(a)). The rationale for these choices is systematically analyzed in our ablation study 4.3. Since each schedule has a different optimal learning rate in practice, to thoroughly address this concern, we conduct additional experiments to evaluate the performance of all schedules with different learning rates in Appendix E.8.

Baselines Research on budgeted-iteration training remains limited, with most existing approaches focusing on learning rate scheduling strategies [5, 30, 44]. To provide a fair comparison, we adopt several widely used and empirically effective learning rate schedules as baselines: **Step(SS)** [16], **Cosine(CS)** [33], **Cyclical (CLR) and OneCycle (1C)** [43, 44], **Budgeted training(BT, mathematically equivalent to linear decay)** [30] and **Reflected Exponential (REX)** [5]. Details of these baselines are provided in Appendix E.1.

4.1 Experiments for vision classification tasks

We evaluate our proposed UBA schedule on vision benchmarks (e.g., CIFAR10/100 and ImageNet) using different architectures (ResNet18, ResNet34, ResNet50). In addition, we independently train models using fixed epoch budgets of 25%, 50%, and 100% of the maximum training epochs, without reusing or interpolating results from longer training runs. This setup ensures that each budget setting is evaluated in isolation, thereby preserving the integrity of comparisons across low and high training budgets. Table 2 presents the validation accuracy across different training budgets, comparing UBA against six baseline schedules (see detailed results in Appendix E.4).

We find that: (i) UBA demonstrates the strongest performance across all training budgets, achieving superior results on both small-scale (CIFAR10/100 with ResNet18/34) and large-scale (ImageNet with ResNet50) benchmarks. This consistent improvement highlights the *generalizability across model and dataset scales*. (ii) UBA outperforms baselines not only at 100% training budget but also at 25% and 50% iteration budgets, demonstrating its *budget efficiency*, i.e. effectiveness in computation-constrained scenarios. (iii)Notably, UBA shows robust performance even while the second-best schedule varies depending on both the datasets, architectures and training budgets. For practitioners seeking reliable schedules without extensive method selection, UBA provides a default-strong choice. UBA eliminates the need for case-by-case baseline comparison and delivers *stable superiority and reliability* regardless of datasets, architectures, training budgets and scales.

Table 2: Validation accuracy for vision classification tasks. We present validation accuracy on vision benchmarks (e.g., CIFAR10/100 and ImageNet) using different architectures (ResNet18, ResNet34, ResNet50) under fixed epoch budgets of 25%, 50%, and 100% of the maximum training epochs.

Dataset	Network	Method	training budget (epoch(%))		
			75 (25%)	150(50%)	300(100%)
CIFAR10	ResNet18	SS	92.71±0.4243	93.45±0.4822	93.61±0.3035
		CS	94.21±0.5116	94.24±0.2419	95.52±0.2433
		CLR	92.13±0.2610	92.99±0.5459	94.92±0.3811
		1C	94.23±0.1838	95.03±0.1353	95.35±0.2984
		BT	94.18±0.3546	94.79±0.4605	95.68±0.1168
		REX	94.49±0.3225	95.10±0.1929	95.45±0.1350
		UBA(ours)	**94.57±0.3281**	**95.26±0.2150**	**95.65±0.1589**
			75 (25%)	150(50%)	300(100%)
CIFAR100	ResNet34	SS	71.09±0.2676	75.34±0.1802	77.24±0.1808
		CS	64.12±0.1808	69.84±0.7580	74.29±0.5958
		CLR	73.33±0.5635	73.41±0.2835	74.74±0.2266
		1C	73.38±0.3227	75.04±0.6382	77.86±0.2239
		BT	72.75±0.2611	75.35±0.2423	78.58±0.2705
		REX	73.64±0.5635	75.14±0.3110	78.04±0.1430
		UBA(ours)	**74.60±0.1508**	**76.63±0.2678**	**78.95±0.2818**
			75 (25%)	150(50%)	300(100%)
ImageNet	ResNet50	SS	74.74±0.1386	77.24±0.3861	78.56±0.6435
		CS	75.84±0.2220	77.95±0.2164	79.07±0.0438
		CLR	73.90±1.4609	76.04±1.6419	77.84±1.3053
		1C	75.83±0.7792	77.74±0.5218	78.90±0.2956
		BT	75.90±0.2786	77.81±0.4624	78.86±0.2814
		REX	75.85±0.8047	77.66±0.8853	78.66 ±0.2828
		UBA(ours)	**76.29±0.4031**	**78.26±0.1640**	**79.39±0.0170**

4.2 Experiments for language models

We evaluate UBA schedule on the OLMo[19], a truly open language model based on decoder-only transformer architecture, across diverse benchmarks in language model evaluation. Since large language models commonly provide multiple scales to accommodate different compute constraints. We adjust both model size and training steps to explore budgets. We evaluate UBA on OLMo networks

spanning four parameter scales: 36M, 73M, 151M, and 300M, covering normal to large-scale models. Due to space limitation, we defer details such as benchmarks introduction, experimental setting and overall results in Appendix E.5).

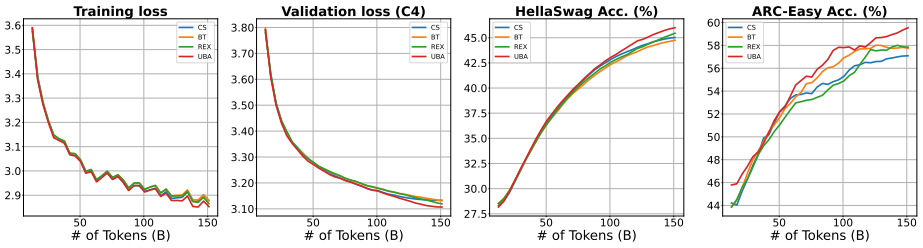


Figure 2: Training dynamics and performance for language tasks under 150B tokens on 300M OLMo. We present the training loss, validation loss, and downstream performance on HSWAG and ARC-E, demonstrating that UBA schedule achieves superior performance.

From Table 3, UBA achieves state-of-the-art performance on approximately 50% of the benchmarks across all scales while the second-best schedule varies. Furthermore, it achieves *consistent superior average performance* among all baselines. It highlights the stable superiority and reliability of UBA, providing a *default-strong choice*. Moreover, it demonstrates significant improvements in SciQ-73M(+1.7) and ARC-E-300M(+2.63), highlighting its ability to enhance generalization across diverse benchmarks. Besides, in Figure 2, UBA consistently achieves lower training loss and validation loss throughout the training, indicating the *efficient training ability* and downstream performance enhancement. Notably, while task-specific tuning of φ can further improve model performance, we intentionally fix its value across all tasks and architectures to ensure fair evaluation. Remarkably, even with this universal φ setting, UBA consistently outperforms baselines on diverse benchmarks, demonstrating *inherent robustness* to varying benchmarks and model scales.

Table 3: Performance comparison between UBA and the best-performing baseline schedules on OLMo. We report the results of top-performing baseline schedule for the corresponding benchmark and model scale. The names of the top-performing baseline schedules are listed below the results. Bold values indicate UBA's superiority (See overall results of each schedule in Appendix E.5).

		Benchmark(accuracy %)										
Size	Sched.	PIQA	HSWAG	OBQA	SciQ	ARC-E	ARC-C	COPA	SIQA	SOC	OTH	Avg.
36M	Best baseline	61.15 (CLR)	27.91 (1C)	27.80 (REX)	68.10 (REX)	45.26 (SS)	23.41 (1C)	63.00 (SS)	41.45 (CLR)	25.18 (REX)	29.86 (1C)	40.92 (REX)
	UBA	60.39	**27.98**	27.40	**68.20**	**45.79**	21.74	**63.00**	40.69	24.33	**30.14**	**40.97**
73M	Best baseline	62.46 (REX)	30.22 (REX)	28.80 (CS)	72.60 (CS)	47.54 (1C)	26.42 (1C)	66.00 (BT)	41.97 (BT)	26.32 (SS)	29.34 (REX)	42.53 (CS)
	UBA	**63.17**	30.09	**28.80**	**74.30**	45.79	22.74	65.00	41.45	**27.13**	28.85	**42.73**
150M	Best baseline	66.00 (CS)	35.72 (REX)	32.80 (1C)	78.20 (BT)	53.16 (1C)	26.42 (BT)	69.00 (REX)	43.71 (CLR)	27.68 (REX)	32.64 (REX)	45.91 (REX)
	UBA	65.23	35.50	29.80	**78.30**	50.35	**27.42**	67.00	43.50	**29.29**	**32.72**	**45.91**
300M	Best baseline	70.18 (REX)	46.30 (REX)	33.40 (CS)	84.40 (REX)	57.72 (REX)	28.43 (REX)	72.00 (BT)	44.58 (CS)	29.50 (REX)	36.74 (REX)	49.70 (REX)
	UBA	69.48	**46.44**	**34.60**	83.90	**60.35**	**29.10**	**72.00**	44.42	28.53	35.41	**50.42**

4.3 Ablation study

Performance across different optimizers UBA originates from the optimization problem under gradient descent dynamics. While modern optimizers (e.g., AdamW [32]) introduce momentum and adaptive mechanisms, they maintain the fundamental property of gradient-based updates. To verify the cross-optimizer performance, we conduct ablation studies between SGD and AdamW

optimizers(see detailed results in Appendix E.6). The results show that UBA achieves SOTAs on both SGD and AdamW optimizers, demonstrating the *cross-optimizer robustness* of UBA. This highlights its *broad applicability* beyond standard gradient descent.

Parameter analysis of φ We perform a sensitivity analysis of $\varphi \in \{0.25, 0.5, 1.0, 2.5, 5, 10\}$ in equation (5) , which controls the variation speed of learning rate (see detailed setting, results and analyses in Appendix E.7).

Figure 11 show AdamW prefers smaller φ while SGD requires larger φ. We attribute this phenomenon to the preconditioning effect of AdamW. As the relationship formalized in Proposition 1), smaller φ is favored for low optimization difficult, while larger φ is beneficial for difficult optimization. AdamW adapts the learning rate by scaling gradients with the second moment estimate $\sqrt{v_t}$, implicitly reducing the condition number of the optimization landscape. Thus the schedule with smaller φ is preferred. In contrast, SGD lacks such adaptive mechanisms, thus a larger φ is more suitable for this situation. This alignment between theory and experiment underscores the importance of tailoring φ to the optimizer's characteristics. *Overall, φ is related to a generalized optimization difficulty, where optimization precondition effect, datasets distribution and network architecture are all related to optimization difficulty.*

Performance across different periods Our schedule has a periodic phase-based learning rate adjustment setting, where the learning rate at the k-th phase is dynamically determined by the k-th local minimum of the loss landscape. To validate our scheduling strategy, we conduct experiments by varying K (see detailed results in Appendix E.9). The results suggests that multi-phase scheduling captures the dynamic features of the loss surface more finely, but it needs careful selection for φ, where φ reflects optimization difficulty. However, selecting optimal φ values per phase for multi-phase remains non-trivial, which motivates future work on automated landscape-aware φ tuning.

5 Conclusion

In this paper, we construct a unified budget-aware training optimization framework that inherently adapts to landscape curvature variations, enhancing training robustness. From this optimization framework, we propose the Unified Budget-Aware (UBA) schedule. Extensive experiments demonstrate UBA *consistently surpasses* the commonly-used schedules across diverse vision and language tasks, spanning network architectures (e.g., ResNet, OLMo) and scales, under different training-iteration budgets. Moreover, our performance improvements come with negligible computational overhead, which constitutes one of its key advantages.

Theoretically and empirically, we observe that the parameter φ correlates with the optimization difficulty of the training process, influences the optimal choice of φ and UBA's performance. However, the explicit relationship between φ and optimization difficulty remains unexplored and no established evaluation metric exists to quantify the optimization difficulty. These limitations motivate future work on optimization difficulty-aware φ tuning.

Despite these open questions, the effectiveness, implementation simplicity and ease of tuning make the UBA a practical, must-try schedule for deep learning practitioners. We hope this work could motivate more studies on learning rate scheduling.

6 Acknowledgments and disclosure of funding

Z. Lin was supported by National Key R&D Program of China (2022ZD0160300), the NSF China (No. 62276004) and the State Key Laboratory of General Artificial Intelligence.

References

[1] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. How important is importance sampling for deep budgeted training? *arXiv preprint arXiv:2110.14283*, 2021.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[4] Tom B Brown. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901, 2020.

[5] John Chen, Cameron Wolfe, and Tasos Kyrillidis. Rex: Revisiting budgeted training with an improved schedule. *Proceedings of Machine Learning and Systems*, 4:64–76, 2022.

[6] Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. Data-efficient gan training beyond (just) augmentations: A lottery ticket perspective. *Advances in Neural Information Processing Systems*, 34:20941–20955, 2021.

[7] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36, 2024.

[8] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[9] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[11] Aaron Defazio, Ashok Cutkosky, Harsh Mehta, and Konstantin Mishchenko. Optimal linear decay learning rate schedules and further refinements. *arXiv preprint arXiv:2310.07831*, 2023.

[12] Simon Du, Jason Lee, Yuandong Tian, Aarti Singh, and Barnabas Poczos. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. In *International Conference on Machine Learning*, pages 1339–1348. PMLR, 2018.

[13] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[14] Donald A. Flanders and George Shortley. Numerical determination of fundamental modes. *Journal of Applied Physics*, 21(12):1326–1332, 1951.

[15] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, and Jeffrey Hsu. A framework for few-shot language model evaluation. December 2023. URL `https://zenodo.org/records/10256836`.

[16] Rong Ge, Sham M Kakade, Rahul Kidambi, and Praneeth Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[17] Jonas Geiping and Tom Goldstein. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pages 11117–11143. PMLR, 2023.

[18] Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. Semeval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, 2012.

[19] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

[20] Alex Hägele, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

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

[22] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[23] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[24] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[25] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

[26] Peter Izsak, Moshe Berchansky, and Omer Levy. How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*, 2021.

[27] Jean Kaddour. Stop wasting my time! saving days of imagenet and bert training with latest weight averaging. *arXiv preprint arXiv:2209.14981*, 2022.

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

[29] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. *arXiv preprint arXiv:1905.04753*, 2019.

[31] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on machine learning*, pages 5958–5968. PMLR, 2020.

[32] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[33] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[34] Kairong Luo, Haodong Wen, Shengding Hu, Zhenbo Sun, Zhiyuan Liu, Maosong Sun, Kaifeng Lyu, and Wenguang Chen. A multi-power law for loss curve prediction across learning rate schedules. In *International Conference on Learning Representations (ICLR)*, 2025.

[35] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

[36] Rui Pan, Haishan Ye, and Tong Zhang. Eigencurve: Optimal learning rate schedule for sgd on quadratic objectives with skewed hessian spectrums. *arXiv preprint arXiv:2110.14109*, 2021.

[37] Rui Pan, Shizhe Diao, Jianlin Chen, and Tong Zhang. Extremebert: A toolkit for accelerating pretraining of customized bert. *arXiv preprint arXiv:2211.17201*, 2022.

[38] Xuran Pan, Xuan Jin, Yuan He, Shiji Song, Gao Huang, et al. Budgeted training for vision transformer. In *The Eleventh International Conference on Learning Representations*, 2022.

[39] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[40] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

[41] Li Shen, Yan Sun, Zhiyuan Yu, Liang Ding, Xinmei Tian, and Dacheng Tao. On efficient training of large-scale deep learning models: A literature review. *arXiv preprint arXiv:2304.03589*, 2023.

[42] Yong Shi, Anda Tang, Lingfeng Niu, and Ruizhi Zhou. Sparse optimization guided pruning for neural networks. *Neurocomputing*, 574:127280, 2024.

[43] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[44] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[45] Vivienne Sze, Yu-Hsin Chen, Joel Emer, Amr Suleiman, and Zhengdong Zhang. Hardware for machine learning: Challenges and opportunities. In *2017 IEEE custom integrated circuits conference (CICC)*, pages 1–8. IEEE, 2017.

[46] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

[47] 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*, pages 10347–10357. PMLR, 2021.

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[50] Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.

[51] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

[52] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[53] Kh Shahriya Zaman, Mamun Bin Ibne Reaz, Sawal Hamid Md Ali, Ahmad Ashrif A Bakar, and Muhammad Enamul Hoque Chowdhury. Custom hardware architectures for deep learning on portable devices: a review. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6068–6088, 2021.

[54] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[55] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[56] Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. Exploring flat minima for domain generalization with large learning rates. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[57] Yushun Zhang, Dmitry Rybin, and Zhi-Quan Luo. Finite horizon optimization: Framework and applications. *arXiv preprint arXiv:2412.21068*, 2024.

[58] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction (e.g., UBA consistently surpasses the commonly-used schedules on vision and language tasks and sets new SoTAs for many networks and frameworks, e.g. ResNet and OLMo, under different training iterations.) are fully aligned with the paper's contributions and scope. We provide an in-depth experiments and analyses of the proposed learning schedule across vision and language tasks, networks, scales and training budgets.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss limitations and future work in Section 5. Besides, the assumptions can be found in Appendix A. We test our method across vision and language tasks, various scales of datasets and networks, under different iteration budgets.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The full set of assumptions are in Appendix A. The complete derivations and proof are in Appendix A and D.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Appendix E, we carefully describe provide complete information needed to reproduce the experimental results, including hyper-parameters(initial learning rates, batch sizes, configurations of optimizer and schedules) for all tasks, implementation versions(such as PyTorch 2.4.1+cu118 on CIFAR), hardware configurations(GPU models, memory), full results of ablation studies (see Appendix E). Code and data are also available.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide open access to the code in the Appendix. Moreover, we provide sufficient instructions to faithfully reproduce the main experimental results in Appendix E.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: These details are described in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the high computational cost of training large-scale models (e.g., Transformers on ImageNet), we report results from a single run with fixed random seeds. To ensure reliability, we list experimental setting and provide code for deterministic reproduction. To maintain uniform evaluation protocols across all experiments (large/small-scale),

we prioritized identical statistical reporting standards. This avoids selective rigor that could mislead comparisons. Besides, we conduct parameter analyses to improve reliability. Moreover, small-scale results align with the expectations and results on large-scale experiments, reducing the need for empirical variance quantification. However, we provide error bars of curve fitting on numerical solution.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computation resource is described in Appendix E. we carefully describe provide complete information including type of compute workers and memory.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conform with the NeurIPS code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses societal impacts in Section 5. It highlights potential benefits, such as improving model performance and offering a practical scheduling strategy for practitioners, which may inspire further research on learning rate scheduling. Since we note that the proposed method is a generic optimization algorithm with no explicit negative societal impacts, indirect risks such as computational costs from sub-optimal hyper-parameter selection are not thoroughly examined. We provide the hyper-parameter selection guideline through analyses to mitigate this indirect risk.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

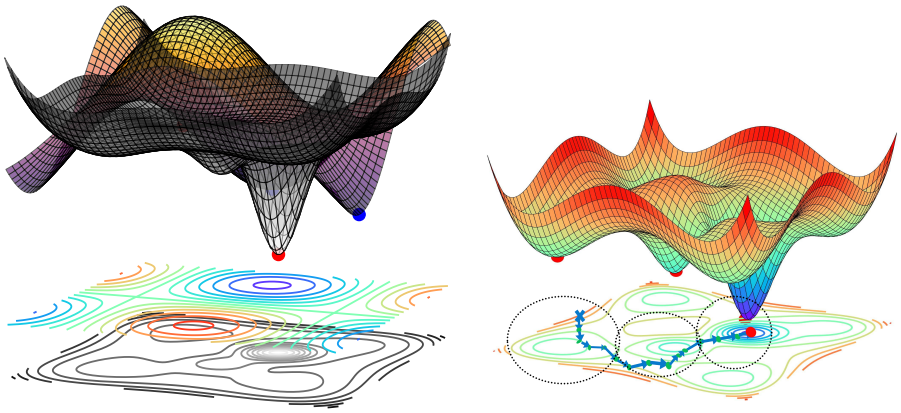
    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Appendix

A Methodology

Conceptual illustration The optimization landscape of neural networks is highly dynamic due to data distribution and sampling, and sensitive to various factors, such as network architecture, optimization and hyper-parameters. Figure 3(a) illustrates the inherent uncertainties during the training process. The black-and-white surfaces reflect the loss landscape processing one batch, while the colored surface represents the landscape following the next batch. The substantial variations in the landscape between consecutive batches significantly affect the training trajectory, making it challenging to design a one-size-fits-all learning rate strategy that can effectively adapt to such fluctuations. Therefore, we aim to develop a strategy a robust approach to be scheduled leaning rate across diverse optimization landscapes. We construct a budget-iteration-aware framework for learning rate optimization model that guarantees minimal loss within constrained training iterations under the worst-case conditions.

We analyze loss landscape by a quadratic approximation around nearby strict local optima. This approach enables us to capture the key features of the loss surface near these points. Therefore, the trajectory of optimization is impacted by the characteristics of the nearby strict local optima, since the optimization process is inherently shaped by the loss landscape and the key features of the surface are captured by these optima, as illustrated in Figure 3(b), Consequently, we can derive the learning rate within the optimization model by the information of these nearby strict local optima.



(a) (b)

Figure 3: Conceptual illustration: visualization of the motivations behind the proposed modeling approaches. (a) shows the rationale for the modeling approach in (1), while (b) explains the modeling approach for (2). The solid line represents the optimization trajectory, while the dashed circles indicate the local approximation around corresponding minima.

Assumptions The optimization behavior of a neural network is fundamentally governed by the geometry of its loss landscape, which emerges from the interaction of parameter configuration, datasets characteristics, sampling, network architecture, optimization preconditioning effect. In the vicinity of any strict local minimum, the loss surface can be approximated by a quadratic method whose curvature is determined by the local Hessian matrix. Thus the optimization trajectory is consequently dominated by the geometric properties (e.g., Hessian features, curvature profiles) of the nearest strict local minima.

Many tasks train models using epochs, where one epoch represents a full pass through the datasets. Under this circumstance, an iteration occurs when the model processes a single batch. While the number of iterations per epoch varies with batch size, training with a fixed epoch budget becomes equivalent to using a fixed iteration count when batch size remains constant. For consistency, this paper uses iterations as our primary unit of optimization model construction and networks training.

The sampling Hessian satisfies: $\mathbb{E}_\xi \left[n_t n_t^\top \right] \preceq \sigma^2 H_f^{(k)}$ for some constant σ. This assumption is followed by the work [16, 36]. Here, σ^2 measures the degree of Hessian inconsistency across data samples, i.e., the variance in the per-sample Hessian matrices. Besides, a smaller σ^2 indicates a more stable curvature landscape between each data sampling, leading to lower noise amplification during optimization.

Optimization problem derivation for Section 3.1 Considering the first constraint $W_{t+1} = W_t - \eta_t H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)})$ in min-max optimization (2), we can reformulate it as

$$W_{t+1} - \bar{W}^{(k)} = (I - \eta_t H_f^{(k)}(\xi))(W_t - \bar{W}^{(k)}), \tag{9}$$

where I denotes the identity matrix. By iteratively applying this equation (9), we obtain

$$\begin{aligned} &(W_{T_{k+1}} - \bar{W}^{(k)}) = \\ &(I - \eta_{T_{k+1}-1} H_f^{(k)}(\xi)) \cdots (I - \eta_{T_k+1} H_f^{(k)}(\xi))(W_{T_k+1} - \bar{W}^{(k)}). \end{aligned} \tag{10}$$

Since the matrix $H_f^{(k)}(\xi)$ is positive semi-definite, it possesses N eigenvalues $\lambda_1^{(k)}(f(\xi)), \lambda_2^{(k)}(f(\xi)), \cdots, \lambda_N^{(k)}(f(\xi))$ abbreviated as $\lambda_i^{(k)}$, which satisfy $0 < \lambda_l^{(k)} \leq \lambda_i^{(k)}(f(\xi)) \leq \lambda_u^{(k)}$ for all i. Here, $\lambda_l^{(k)}$ and $\lambda_u^{(k)}$ denote the bounds on the non-zero eigenvalues of the Hessian around k-th optimum. It also satisfies $\lambda_u^{(k)} \leq \frac{2}{\eta}$ where η is the learning rate to guarantee the convergence [56]. Moreover, there exist N linearly independent eigenvectors $u_1^{(k)}(f(\xi)), u_2^{(k)}(f(\xi)), \cdots, u_N^{(k)}(f(\xi))$ abbreviated as $u_i^{(k)}$, which can form the basis for the N dimension vector space. Then we have $H_f^{(k)}(\xi) u_i^{(k)} = \lambda_i^{(k)} u_i^{(k)} \quad (i = 1, 2, \cdots, N)$ and the initial deviation from the optimal solution can be expressed as $W_{T_k} - \bar{W}^{(k)} = \sum_{i=1}^{N} s_i^{(k)} u_i^{(k)}$ where $s_i^{(k)}$ are the coefficients corresponding to the eigenvector components. Thus, we obtain

$$\begin{aligned} \left\| W_{T_{k+1}} - \bar{W}^{(k)} \right\|_2^2 &= \sum_{i=1}^{N} (s_i^{(k)})^2 \|u_i^{(k)}\|_2^2 \left[\prod_{t=1+T_k}^{T_{k+1}} (1 - \eta_t \lambda_i^{(k)}) \right]^2 \\ &\leq \sum_{i=1}^{N} (s_i^{(k)})^2 \|u_i^{(k)}\|_2^2 \max_{\lambda_l^{(k)} \leq \lambda_i^{(k)} \leq \lambda_u^{(k)}} \left[\prod_{t=1+T_k}^{T_{k+1}} (1 - \eta_t \lambda_i^{(k)}) \right]^2 \\ &= \max_{\lambda_l^{(k)} \leq \lambda_i^{(k)} \leq \lambda_u^{(k)}} \left[\prod_{t=1+T_k}^{T_{k+1}} (1 - \eta_t \lambda_i^{(k)}) \right]^2 \left\| W_{T_k} - \bar{W}^{(k)} \right\|_2^2 \end{aligned} \tag{11}$$

B Numerical solution and curve fitting

To solve the constrained min-max problem (4) as follow,

$$\min_{\eta_{1+T_k},\eta_{2+T_k},\ldots,\eta_{T_{k+1}}} \max_{\lambda_l^{(k)}\le\lambda_i^{(k)}\le\lambda_u^{(k)}} \prod_{t=1+T_k}^{T_{k+1}} \left[(1-\eta_t\lambda_i^{(k)})\right]^2$$
$$s.t. \quad \eta_{1+T_k},\eta_{2+T_k},\ldots,\eta_{T_{k+1}} \in [\eta_{\min},\eta_{\max}]$$
$$k=1,2,\ldots K$$

we adopt an iterative projected gradient method that alternates between minimizing over the variables η_t and maximizing over the parameters $\lambda_i^{(k)}$. Specifically, at each iteration, we update η_t with fixed $\lambda_i^{(k)}$ as follows

$$\eta_t \leftarrow \mathcal{P}_{[\eta_{\min},\eta_{\max}]}\left(\eta_t - \alpha\nabla_{\eta_j}\mathcal{L}(\eta_t,\lambda_i^{(k)})\right).$$

For fixed η_t, we update $\lambda_i^{(k)}$ via

$$\lambda_i^{(k)} \leftarrow \mathcal{P}_{[\lambda_l,\lambda_u]}\left(\lambda_i^{(k)} + \beta\nabla_{\lambda_i^{(k)}}\mathcal{L}(\eta_t,\lambda_i^{(k)})\right),$$

where $\mathcal{P}$ is the projection over box constraints, α and β are step sizes.

This approach requires per-network numerical optimization to determine optimal learning rates case-by-case. To circumvent repeated numerical optimization for learning rate scheduling whenever a new network is trained, we propose to fit these solutions with a parametric function universally across networks.

We note that the solution of optimization model is an unordered set $\{\eta_{1+T_k},\eta_{2+T_k},\ldots,\eta_{T_{k+1}}\}$, meaning that any permutation of these values yields the same objective value. To facilitate function approximation and improve the training stability of the network, we sort the solution set in both descending and ascending orders before fitting them with a smooth function. This pre-processing step preserves the optimality of the solution while enhancing the stability and interpretability of the fitted function.

Due to the symmetry between the descending and ascending orders, and without loss of generality, we first fit a function to the descending order solution. The corresponding function for the ascending order can then be easily obtained via a simple transformation.

Our key observation is that the numerical solutions under varying λ_l, λ_u configurations exhibit two characteristic decay modes. As shown in Figure 4, these transformations manifest through two distinct mechanisms: (i) gradual-to-accelerated decay: gentle initial decrease transitioning to rapid drop (as shown in Figures. 4(d),4(e) and 4(f)); (ii) rapid-to-gradual decay: Initial steep decline followed by slow decay (as shown in Figures. 4(g),4(h) and 4(i)). These pattern can be formulated as transformations of a base function, which resembles the curvature variation of the transformed ℓ_1 norm modulated by a parameter [42].

We first isolate the base function by examining the limiting case where $\lambda_l \approx \lambda_u$. In this regime, the numerical solution within each interval $[1+T_k, T_{k+1}]$ manifests a cosine-like profile (Figures. 4(a),4(b) and 4(c)), suggesting the base form:

$$1+\cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)})$$

Then we construct the fitting function as:

$$(\eta_{\max}-\eta_{\min})\frac{a(1+\cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}))}{b+c(1+\cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)})} + \eta_{\min}. \tag{12}$$

where $\{a,b,c\}$ control learning rate decay modes.

In addition, through systematic fitting across nine configurations (Table 4), we discover a universal scaling relationship among the parameters:

$$\frac{b+2c}{a} \approx 2 \tag{13}$$

Table 4: The parameter details of curve fitting.

hyper-parameters				fitted parameters				objective value(log)(4)	
$\eta_{\min}$	$\eta_{\max}$	$\lambda_l^{(k)}$	$\lambda_u^{(k)}$	a	b	c	φ	by numerical solution	by fitting function
0	10	1.89	2.60	0.26	0.51	0.00	1.99	792.67	750.83
0	10	2.39	2.61	4.16	7.35	0.53	1.80	798.38	770.38
0	10	5.29	5.61	4.84	7.32	1.28	1.56	1118.79	1105.80
0	10	18.89	260.11	1124.67	212.28	1011.94	0.19	2933.34	2933.62
0	5	28.89	200.11	755.91	85.01	719.40	0.12	2607.11	2591.51
0	2	50.89	150.11	0.27	0.05	0.24	0.17	2031.66	2075.50
1	10	10.01	150.11	77.45	684.15	-268.20	8.40	2373.58	2431.42
2	10	18.89	50.11	0.11	2.52	-1.15	21.02	2007.68	2044.84
3	10	20.01	180.11	0.07	3.07	-1.47	38.34	2604.98	2632.21

Table 4 lists the fitted parameters $\{a, b, c\}$, where the mean absolute error is 0.04. Figure 5 visualizes the variability of the $\frac{b+2c}{a}$ by shading the region within ± 1 standard deviation from the theoretical value in Figure 5. The empirical results falls within $\pm 1\sigma$ of the predicted value (2.0 ± 0.04) in most of cases, demonstrating strong agreement with the assumption relation (13).

This empirical relationship (13) allows us to reduce the original 3-parameter system to 2 degrees of freedom as follows,

$$(\eta_{\max} - \eta_{\min}) \frac{2a(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}))}{2b + (2a-b)(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)})} + \eta_{\min}. \tag{14}$$

Actually, it is a 1-parameter system

$$(\eta_{\max} - \eta_{\min}) \frac{2(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}))}{2d + (2-d)(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)})} + \eta_{\min}, \tag{15}$$

where $\varphi := b/a$. We list the value of fitted parameter φ in Table 4. Since the equation (14) and the equation (15) are mathematically equivalent, we adopt the equation (15) as the canonical representation throughout this work.

Then the corresponding function for the ascending order can then be easily obtained via a phase shift of π as follow,

$$(\eta_{\max} - \eta_{\min}) \frac{2(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)} + \pi))}{2\varphi + (2-\varphi)(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)} + \pi)} + \eta_{\min}, \tag{16}$$

In summary, we obtain the η_t within the interval $t \in [1 + T_k, T_{k+1}]$ as follows

$$\eta_t = (\eta_{\max} - \eta_{\min}) \frac{2(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)} + (k-1)\pi))}{2\varphi + (2-\varphi)(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)} + (k-1)\pi)} + \eta_{\min}, \tag{17}$$

where d is the hyper-parameter controlling the variation speed of η_t.

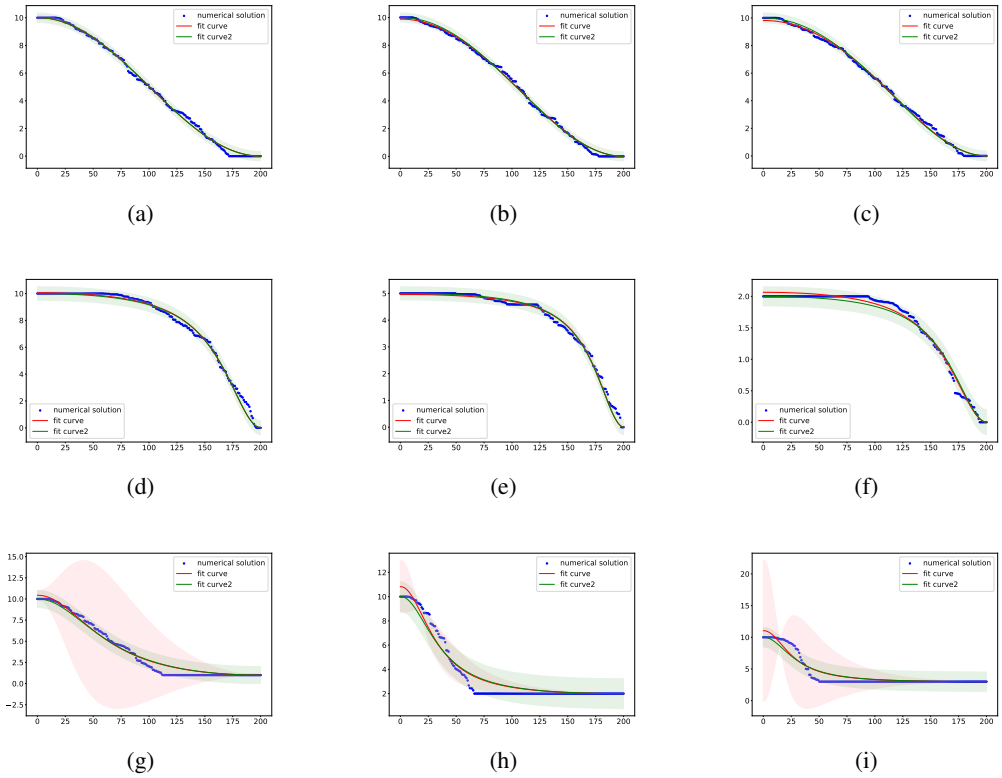


Figure 4: The curve fitting of numerical solutions.

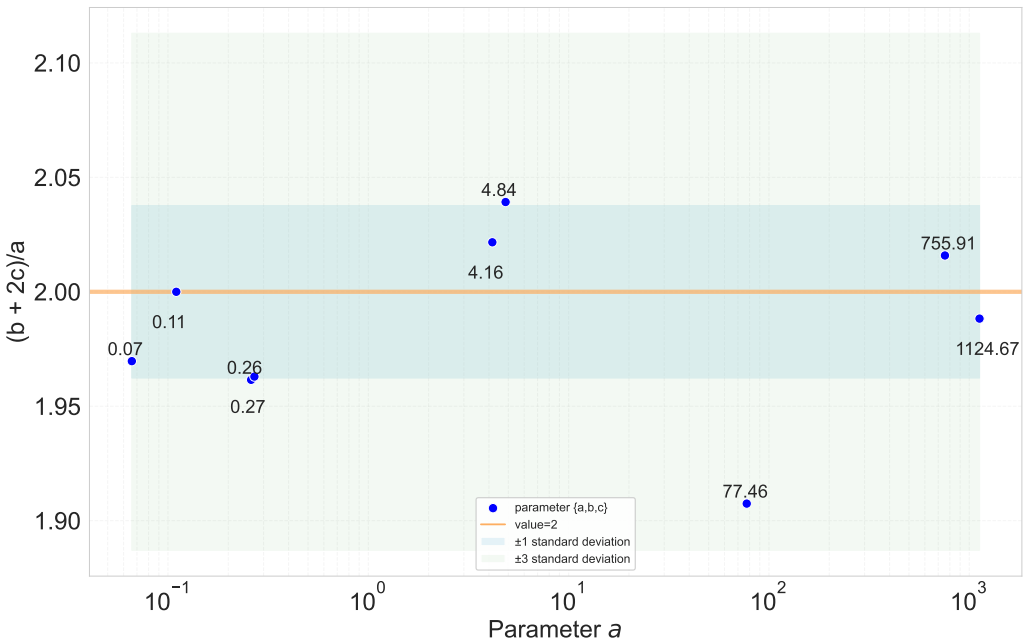


Figure 5: Visualization of relation between parameter a, b and c.

C Adaptive simulation of existing schedules

Our schedule has adaptability owing to the parameter φ, enabling it to approximate the behavior of existing schedules (e.g.step decay, cosine annealing, cyclic schedule or Rex schedule) through simple parameter adjustments. More importantly, it outperforms these schedules by jointly optimizing training stability and final accuracy.

We visualize the adaptive simulation of in Figure 6. By setting our parameter φ to 2, the UBA schedule degenerates to a cosine schedule (figure 6(a)). By setting φ to large value such as 30, the UBA schedule mimics exponential schedule (figure 6(b)). By setting φ to small value such as 0.8, the UBA schedule mimics Rex schedule (figure 6(c)). By setting φ to 0 , the proposed schedule degenerates to a constant. Therefore, we can control $\eta_{\max}$ as piece-wise value depending on iteration, and the UBA schedule degenerates to a step schedule (figure 6(d)). By setting k to any integer larger than 1, the UBA schedule possesses cyclic characters, thus it mimics cyclic schedule (figure 6(e) 6(f)). Since the OneCycle learning rate schedule can be viewed as a single-period version of cyclical learning rates, the UBA schedule mimics the OneCycle approach by emulating the cyclical schedule.

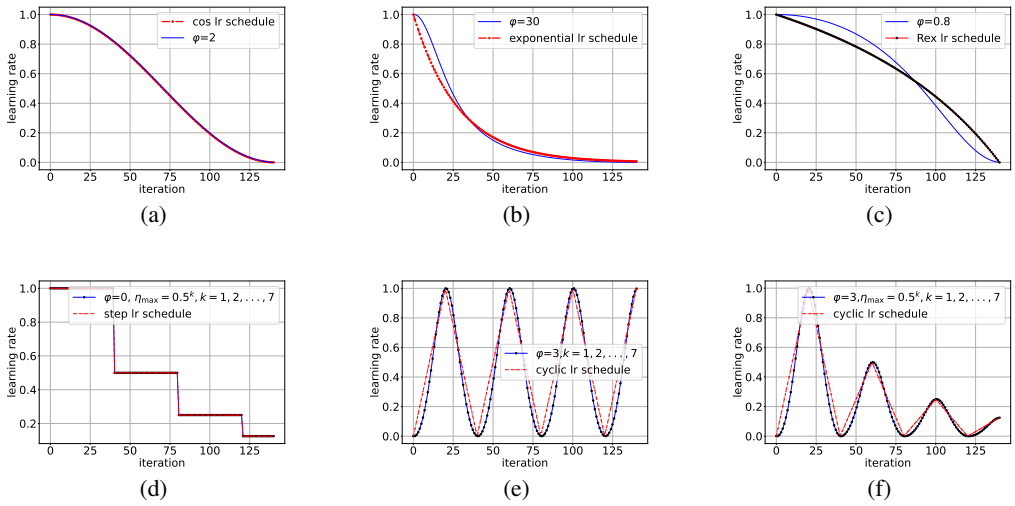


Figure 6: Adaptive simulation by UBA schedule.

D Propositions and lemmas

D.1 Proof of Proposition 1

Proposition 1 *The fitted function* (5) *is the exact closed-form solution* $\frac{2(1+\cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}))}{2\varphi+(2-\varphi)(1+\cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}))}$ *to the min-max optimization problem:*

$$\min_{\eta_{1+T_k},\eta_{2+T_k},\ldots,\eta_{T_{k+1}}} \max_{\lambda_l^{(k)}\le\lambda_i^{(k)}\le\lambda_u^{(k)}} \prod_{t=1+T_k}^{T_{k+1}} \left[(1-\left((\frac{1}{\lambda_l^{(k)}}-\frac{1}{\lambda_u^{(k)}})\eta_t+\frac{1}{\lambda_u^{(k)}}\right)\lambda_i^{(k)})\right]^2 \tag{18}$$

when the hyper-parameter φ *are determined by* $\lambda_l^{(k)}$ *and* $\lambda_u^{(k)}$ *through the relation* $\varphi=2\frac{\lambda_u^{(k)}}{\lambda_l^{(k)}}$ *and* $\eta_{\max}=1, \quad \eta_{\min}=0.$

Proof. According to the theorem in [14], among all polynomials of degree n in μ that take the value +1 at $\mu=d>1$, the Chebyshev polynomial $S_n(\mu)=\frac{C_n(\mu)}{C_n(d)}$ is the unique solution that minimizes the maximum absolute value over the interval $(-1,+1)$. Here, $C_n(\mu)=\cos(n\arccos(\mu))$ is the n-th order Chebyshev polynomial.

We define the polynomial $Q(\lambda_i^{(k)})$ as:

$$Q(\lambda_i^{(k)}) := \prod_{t=T_k+1}^{T_{k+1}} \left[1-\left(\left(\frac{1}{\lambda_l^{(k)}}-\frac{1}{\lambda_u^{(k)}}\right)\eta_t+\frac{1}{\lambda_u^{(k)}}\right)\lambda_i^{(k)}\right]$$

which is a degree $n=T_{k+1}-T_k$ polynomial in $\lambda_i^{(k)}$.

Next, we introduce the variable transformation:

$$\gamma=\frac{-2\lambda_i^{(k)}+\lambda_l^{(k)}+\lambda_u^{(k)}}{\lambda_u^{(k)}-\lambda_l^{(k)}}$$

which maps the interval $\lambda_l^{(k)}\le\lambda_i^{(k)}\le\lambda_u^{(k)}$ to the interval $-1\le\gamma\le 1$, such that $\lambda_i^{(k)}=\lambda_u^{(k)}$ corresponds to $\gamma=-1$ and $\lambda_i^{(k)}=\lambda_l^{(k)}$ corresponds to $\gamma=1$.

The polynomial $P(\gamma)=Q(\lambda_i^{(k)})$ now satisfies the condition in the theorem from [14], i.e.,

$$P\left(\frac{\lambda_l^{(k)}+\lambda_u^{(k)}}{\lambda_u^{(k)}-\lambda_l^{(k)}}\right)=1 \quad \text{since} \quad Q(0)=1.$$

Therefore, the original min-max problem

$$\min_{\eta_{T_k+1},\eta_{T_k+2},\ldots,\eta_{T_{k+1}}} \max_{\lambda_l^{(k)}\le\lambda_i^{(k)}\le\lambda_u^{(k)}} \prod_{t=T_k+1}^{T_{k+1}} \left[1-\left(\left(\frac{1}{\lambda_l^{(k)}}-\frac{1}{\lambda_u^{(k)}}\right)\eta_t+\frac{1}{\lambda_u^{(k)}}\right)\lambda_i^{(k)}\right]^2$$

is equivalent to the problem of finding a polynomial in γ of degree $T_{k+1}-T_k$ that minimizes the maximum absolute value in the interval $-1\le\gamma\le 1$.

By the theorem in [14], the desired solution to the problem is the Chebyshev polynomial:

$$S_{T_{k+1}-T_k}(\gamma)=\frac{C_{T_{k+1}-T_k}(\gamma)}{C_{T_{k+1}-T_k}\left(\frac{\lambda_l^{(k)}+\lambda_u^{(k)}}{\lambda_u^{(k)}-\lambda_l^{(k)}}\right)}$$

To match $P(\gamma)=S_{T_{k+1}-T_k}(\gamma)$, we equate the corresponding zeros. The roots of $Q(\lambda_i^{(k)})=0$ are given by:

$$\lambda_i^{(k)}=\frac{1}{\left(\left(\frac{1}{\lambda_l^{(k)}}-\frac{1}{\lambda_u^{(k)}}\right)\eta_t+\frac{1}{\lambda_u^{(k)}}\right)} \quad \text{for} \quad t=1,2,\ldots,T_{k+1}-T_k.$$

On the other hand, the zeros of $P(\gamma)$ correspond to the values of γ where the polynomial $P(\gamma)$ vanishes, and these zeros are given by:

$$\gamma_t = \frac{\lambda_l^{(k)} + \lambda_u^{(k)}}{\lambda_u^{(k)} - \lambda_l^{(k)}} - \frac{2}{\left(\left(\frac{1}{\lambda_l^{(k)}} - \frac{1}{\lambda_u^{(k)}}\right)\eta_t + \frac{1}{\lambda_u^{(k)}}\right)(\lambda_u^{(k)} - \lambda_l^{(k)})} \quad \text{for} \quad t = 1, 2, \ldots, T_{k+1} - T_k.$$

The zeros of the Chebyshev polynomial $S_{T_{k+1}-T_k}(\gamma)$ are given by the values:

$$\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right) \quad \text{for} \quad t = 1, 2, \ldots, T_{k+1} - T_k.$$

Equating the zeros of $S_{T_{k+1}-T_k}(\gamma)$ and $P(\gamma)$, we have:

$$\frac{\lambda_l^{(k)} + \lambda_u^{(k)}}{\lambda_u^{(k)} - \lambda_l^{(k)}} - \frac{2}{\left(\left(\frac{1}{\lambda_l^{(k)}} - \frac{1}{\lambda_u^{(k)}}\right)\eta_t + \frac{1}{\lambda_u^{(k)}}\right)(\lambda_u^{(k)} - \lambda_l^{(k)})} = \cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)$$

Solving this equation for η_t, we obtain:

$$\eta_t = \frac{(1 + \cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right))}{2\frac{\lambda_u^{(k)}}{\lambda_l^{(k)}} - \left(\frac{\lambda_u^{(k)}}{\lambda_l^{(k)}} - 1\right)\left(1 + \cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}.$$

□

D.2 Proof of Proposition 2

Proposition 2 *Consider the training process within the interval $t \in [1 + T_k, T_{k+1}]$. When the hyper-parameter φ is set sufficiently close to 2, the proposed learning rate scheduling formula reduces to the cosine learning rate schedule.*

Proof.

$$\begin{aligned}
&\lim_{\varphi \to 2}(\eta_{\max} - \eta_{\min})\frac{2(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}))}{2\varphi + (2-\varphi)(1 + \cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)})} + \eta_{\min} \\
&= (\eta_{\max} - \eta_{\min})(\frac{1}{2} + \frac{1}{2}\cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)})) + \eta_{\min} \\
&= \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\frac{(t-T_k)\pi}{T_{k+1}-T_k})\cos(\frac{\pi}{2(T_{k+1}-T_k)}) + \sin(\frac{(t-T_k)\pi}{T_{k+1}-T_k})\sin(\frac{\pi}{2(T_{k+1}-T_k)})) + \eta_{\min} \\
&\approx \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\frac{(t-T_k)\pi}{T_{k+1}-T_k})) + \eta_{\min}
\end{aligned} \tag{19}$$

The final approximation holds because each training phase typically contains many iterations, i.e. , making $\frac{\pi}{T_{k+1}-T_k} \approx 0$. Consequently, $\sin(\frac{\pi}{2(T_{k+1}-T_k)}) \approx 0$ and $\cos(\frac{\pi}{2(T_{k+1}-T_k)}) \approx 1$. □

D.3 Proof of Theorem 1

Lemma 1. *Let η_j be the learning rate at the iteration $j \in [1 + T_k, t]$, defined by the proposed form (5). Let $\lambda_i^{(k)}$ be the eigenvalues of the Hessian matrix $H_f^{(k)}(\xi)$ at the k-th strict minimum $\bar{W}^{(k)}$. Then for any $t \in [1 + T_k, T_{k+1}]$, the following inequalities hold,*

Case 1 ($\varphi > 2$):

$$\begin{aligned}
&\prod_{j=1+T_k}^{t}\Big[(1-\eta_j\lambda_i^{(k)})\Big]^2 \\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big) \\
&\quad\cdot\Big(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(\varphi-2)\pi}}
\end{aligned} \tag{20}$$

Case 2 ($\varphi < 2$):

$$\begin{aligned}
&\prod_{j=1+T_k}^{t}\Big[(1-\eta_j\lambda_i^{(k)})\Big]^2 \\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big) \\
&\quad\cdot\Big(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(2-\varphi)\pi}}
\end{aligned} \tag{21}$$

Proof. By the fact that $1-x \leq \exp(-x)$, we have

$$\prod_{j=1+T_k}^{t}\Big[(1-\eta_j\lambda_i^{(k)})\Big]^2 \leq \exp\left(-2\sum_{j=1+T_k}^{t}\eta_j\lambda_i^{(k)}\right) \tag{22}$$

Substituting the expression for η_j by the proposed form (5) yields:

$$\begin{aligned}
&\exp\left(-2\sum_{j=1+T_k}^{t}\eta_j\lambda_i^{(k)}\right) \\
&= \exp\left(-2\lambda_i^{(k)}\sum_{j=1+T_k}^{t}(\eta_{\max}-\eta_{\min})\frac{2(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right))}{2\varphi+(2-\varphi)(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)}+\eta_{\min}\right) \\
&= \exp\left(-2\lambda_i^{(k)}\eta_{\min}\left(t-T_k\right)\right)\exp\left(-2\lambda_i^{(k)}\sum_{j=1+T_k}^{t}(\eta_{\max}-\eta_{\min})\frac{2\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}{2\varphi+(2-\varphi)\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}\right)
\end{aligned} \tag{23}$$

The behavior of this term depends on the value of the parameter φ. We consider the following cases:

- if $\varphi \geq 2$, $2-\varphi \leq 0$

  Since

$$\begin{aligned}
&\cos(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}) \geq \cos(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}) \geq 1-\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right) \\
&\geq 1-\pi\left(\frac{(j-T_k)}{(T_{k+1}-T_k)}\right),
\end{aligned} \tag{24}$$

we have

$$\begin{aligned}
&\sum_{j=1+T_k}^{t} \frac{1}{2\varphi + (2-\varphi)\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)} \\
&\leq \sum_{j=1+T_k}^{t} \frac{1}{2\varphi + (2-\varphi)\left(1+1-\pi\left(\frac{(j-T_k)}{(T_{k+1}-T_k)}\right)\right)} \\
&= \sum_{j=1+T_k}^{t} \frac{1}{4 + \frac{(\varphi-2)\pi}{(T_{k+1}-T_k)}(j-T_k)}
\end{aligned} \tag{25}$$

Then the equality (23) becomes

$$\begin{aligned}
&\exp\left(-2\lambda_i^{(k)}\eta_{\min}\left(t-T_k\right)\right)\exp\left(-2\lambda_i^{(k)}\sum_{j=1+T_k}^{t}(\eta_{\max}-\eta_{\min})\frac{2\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}{2\varphi+(2-\varphi)\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}\right) \\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\exp\Bigg[-4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right) \\
&\quad \sum_{j=1+T_k}^{t}\frac{1}{2\varphi+(2-\varphi)\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}\Bigg] \\
&\leq \exp\left(-2\lambda_i^{(k)}\eta_{\min}\left(t-T_k\right)\right)\exp\Bigg[-4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right) \\
&\quad \sum_{j=1+T_k}^{t}\frac{1}{4+\frac{(\varphi-2)\pi}{(T_{k+1}-T_k)}(j-T_k)}\Bigg]
\end{aligned} \tag{26}$$

Note that the function $h(j) := \frac{1}{4+\frac{(\varphi-2)\pi}{T_{k+1}-T_k}(j-T_k)}$ is monotone decreasing over $j \in [1+T_k, t]$, so we can bound the summation from below by the integral:

$$\begin{aligned}
&\sum_{j=1+T_k}^{t} \frac{1}{4 + \frac{(\varphi-2)\pi}{T_{k+1}-T_k}(j-T_k)} \\
&\geq \int_{1+T_k}^{t+1} \frac{1}{4 + \frac{(\varphi-2)\pi}{T_{k+1}-T_k}(j-T_k)}dj \\
&\geq \int_{1+T_k}^{t} \frac{1}{4 + \frac{(\varphi-2)\pi}{T_{k+1}-T_k}(j-T_k)}dj \\
&= \frac{(T_{k+1}-T_k)}{(\varphi-2)\pi}\ln\left(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}{4(T_{k+1}-T_k)+(\varphi-2)\pi}\right)
\end{aligned} \tag{27}$$

Thus the inequality (26) becomes

$$
\begin{aligned}
&\exp\left(-2\lambda_i^{(k)}\eta_{\min}\left(t-T_k\right)\right)\exp\Bigg[-4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)\\
&\sum_{j=1+T_k}^{t}\frac{1}{4+\frac{(\varphi-2)\pi}{(T_{k+1}-T_k)}(j-T_k)}\Bigg]\\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\exp\Bigg[4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)\\
&\frac{(T_{k+1}-T_k)}{(\varphi-2)\pi}\ln\left(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\right)\Bigg]\\
&=\exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\Big(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(\varphi-2)\pi}}\\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\Big(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(\varphi-2)\pi}}
\end{aligned}
\tag{28}
$$

The last inequality is because $\lambda_l^{(k)}\leq\lambda_i^{(k)}$ and $\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\leq 1$.

- if $\varphi\leq 2$, $2-\varphi\geq 0$ Since

$$
1+\cos(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)})\leq\pi-\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}. \tag{29}
$$

We obtain

$$
\begin{aligned}
&\exp\left(-2\lambda_i^{(k)}\eta_{\min}\left(t-T_k\right)\right)\exp\left(-2\lambda_i^{(k)}\sum_{j=1+T_k}^{t}(\eta_{\max}-\eta_{\min})\frac{2\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}{2\varphi+(2-\varphi)\left(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)}\right)\\
&\leq\exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\exp\Bigg[-4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)\\
&\sum_{j=1+T_k}^{t}\frac{1}{2\varphi+(2-\varphi)(1+\cos\left(\frac{(2(j-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)}\Bigg]\\
&\leq\exp\left(-2\lambda_i^{(k)}\eta_{\min}\left(t-T_k\right)\right)\exp\Bigg[-4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)\\
&\sum_{j=1+T_k}^{t}\left(\frac{1}{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(j-T_k-0.5)}\right)\Bigg]
\end{aligned}
\tag{30}
$$

because function $h(j) := \frac{1}{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(j-T_k-0.5)}$ is monotone increasing in the range $[1+T_k, +\infty)$, then it holds that

$$\begin{aligned}
&\sum_{j=1+T_k}^{t} \frac{1}{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(j-T_k-0.5)} \\
&\geq \int_{T_k}^{t} \frac{1}{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(j-T_k-0.5)} dj \\
&= -\frac{(T_{k+1}-T_k)}{(2-\varphi)\pi} \ln\left(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\right)
\end{aligned} \tag{31}$$

Thus, we have

$$\begin{aligned}
&\exp\left(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\right)\exp\Bigg[-4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right) \\
&\sum_{j=1+T_k}^{t}\left(\frac{1}{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(j-T_k-0.5)}\right)\Bigg] \\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\exp\Bigg[4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right) \\
&\frac{(T_{k+1}-T_k)}{(2-\varphi)\pi}\ln\left(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\right)\Bigg] \\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big) \\
&\cdot\Big(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(2-\varphi)\pi}} \\
&\leq \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big) \\
&\cdot\Big(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(2-\varphi)\pi}}
\end{aligned} \tag{32}$$

The last inequality is because $\lambda_l^{(k)} \leq \lambda_i^{(k)}$ and $\Big(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\Big) \leq 1$.

□

Theorem 1 *Let $n_t = H_f^{(k)}(W_t - \bar{W}^{(k)}) - H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)})$ be the stochastic curvature noise introduced by sampling at iteration t. Assume that the sampling Hessian satisfies: $\mathbb{E}_\xi\left[n_t n_t^\top\right] \preceq \sigma^2 H_f^{(k)}$ for some constant σ.*

- *If we set the learning rate as the proposed form* (5) *where hyper-parameter $\varphi > 2$, the loss uncertainty introduced by stochastic gradient method within the interval $t \in [1+T_k, T_{k+1}]$*

can be quantified as two terms, bias term and variance term, bounded as follows,

$$
\begin{aligned}
&\mathbb{E}\Big[\mathcal{L}(f(W_t,\xi)) - \mathcal{L}(f(\bar{W}^{(k)},\xi))\Big] \\
&\leq \Big(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(\varphi-2)\pi}} \\
&\quad \cdot \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\lambda_u^{(k)}\|W_{1+T_k}-\bar{W}^{(k)}\|_2^2 \\
&+ \sigma^2 \sum_{i=_{1+T_k}}^{t} \eta_i^2 \sum_{j=1}^{N} (\lambda_j^{(k)})^2 \exp\Big(-2\lambda_j^{(k)}\eta_{\min}(t-i)\Big) \\
&\quad \cdot \Big(\frac{4(T_{k+1}-T_k)+(\varphi-2)\pi(i-T_k)}{4(T_{k+1}-T_k)+(\varphi-2)\pi(t-T_k)}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(\varphi-2)\pi}}
\end{aligned} \tag{33}
$$

- *If we set the learning rate as the proposed form* (5) *where hyper-parameter $\varphi < 2$, the loss uncertainty introduced by stochastic gradient method within the interval $t \in [1+T_k, T_{k+1}]$ can be quantified as two terms, bias term and variance term, bounded as follows,*

$$
\begin{aligned}
&\mathbb{E}\Big[\mathcal{L}(f(W_t,\xi)) - \mathcal{L}(f(\bar{W}^{(k)},\xi))\Big] \\
&\leq \Big(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)+\frac{(2-\varphi)\pi}{2(T_{k+1}-T_k)}}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(2-\varphi)\pi}} \\
&\quad \cdot \exp\Big(-2\lambda_i^{(k)}\eta_{\min}(t-T_k)\Big)\lambda_u^{(k)}\|W_{1+T_k}-\bar{W}^{(k)}\|_2^2 \\
&+ \leq \sigma^2 \sum_{i=_{1+T_k}}^{t} \eta_i^2 \sum_{j=1}^{N} (\lambda_j^{(k)})^2 \exp\Big(-2\lambda_j^{(k)}\eta_{\min}(t-i)\Big) \\
&\quad \cdot \Big(\frac{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(t-T_k-0.5)}{(2\varphi+2\pi-\varphi\pi)-\frac{(2-\varphi)\pi}{(T_{k+1}-T_k)}(i-T_k-0.5)}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max}-\eta_{\min})\left(1+\cos\left(\frac{(2(t-T_k)-1)\pi}{2(T_{k+1}-T_k)}\right)\right)(T_{k+1}-T_k)}{(2-\varphi)\pi}}
\end{aligned} \tag{34}
$$

Proof. Let $H_f^{(k)}$ denote full batch Hessian at k-th minimum, such that $\mathbb{E}(H_f^{(k)}(\xi)) = H_f^{(k)}$, and denote $n_t = H_f^{(k)}(W_t - \bar{W}^{(k)}) - H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)})$, which indicates the bias. The uncertainty brought by stochastic gradient method is measured as follows,

$$
\begin{aligned}
W_{t+1} - \bar{W}^{(k)} &= W_t - \bar{W}^{(k)} - \eta_t H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)}) \\
&= W_t - \bar{W}^{(k)} - \eta_t H_f^{(k)}(W_t - \bar{W}^{(k)}) \\
&\quad + \eta_t H_f^{(k)}(W_t - \bar{W}^{(k)}) - \eta_t H_f^{(k)}(\xi)(W_t - \bar{W}^{(k)}) \\
&= (I - \eta_t H_f^{(k)})(W_t - \bar{W}^{(k)}) + \eta_t n_t \\
&= (I - \eta_t H_f^{(k)})(I - \eta_{t-1} H_f^{(k)}) \cdots (I - \eta_{t-q} H_f^{(k)})(W_{t-q} - \bar{W}^{(k)}) \\
&\quad + \sum_{i=t-q}^{t} (I - \eta_t H_f^{(k)})(I - \eta_{t-1} H_f^{(k)}) \cdots (I - \eta_{i+1} H_f^{(k)}) \eta_i n_i \\
&:= P_t^{(k)} P_{t-1}^{(k)} \cdots P_{t-q}^{(k)} (W_{t-q} - \bar{W}^{(k)}) + \sum_{i=t-q}^{t} P_t^{(k)} P_{t-1}^{(k)} \cdots P_{i+1}^{(k)} \eta_i n_i
\end{aligned} \tag{35}
$$

where $P_i^{(k)} := (I - \eta_i H_f^{(k)})$.

According to the approximation approach of the objective function (1) in Section 3.1 , i.e. $\mathcal{L}(f(W,\xi)) \approx \mathcal{L}(f(\bar{W}^{(k)},\xi)) + \frac{1}{2}(W-\bar{W}^{(k)})^\top H_f^{(k)}(\xi)(W-\bar{W}^{(k)})$, the loss uncertainty introduced by stochastic gradient method within the interval $t \in [1+T_k, T_{k+1}]$ is quantified as $\mathbb{E}[(W_{T_{k+1}}-\bar{W}^{(k)})^\top H_f^{(k)}(W_{T_{k+1}}-\bar{W}^{(k)})]$, where the loss function is approximated by leveraging the full batch Hessian.

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{L}(f(W_t,\xi)) - \mathcal{L}(f(\bar{W}^{(k)},\xi))\right] = \mathbb{E}\left[(W_t-\bar{W}^{(k)})^\top H_f^{(k)}(W_t-\bar{W}^{(k)})\right] \\
&= \mathbb{E}\Bigg[\left(P_t^{(k)}P_{t-1}^{(k)}\cdots P_{1+T_k}^{(k)}(W_{1+T_k}-\bar{W}^{(k)}) + \sum_{i=1+T_k}^{t} P_t^{(k)}P_{T_{k+1}-1}^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)^\top \\
&H_f^{(k)}\left(P_t^{(k)}P_{t-1}^{(k)}\cdots P_{1+T_k}^{(k)}(W_{1+T_k}-\bar{W}^{(k)}) + \sum_{i=1+T_k}^{t} P_t^{(k)}P_{t-1}^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)\Bigg] \\
&= \mathbb{E}\left[(W_t-\bar{W}^{(k)})^\top P_{1+T_k}^{(k)}\cdots P_t^{(k)}H_f^{(k)}P_t^{(k)}\cdots P_{1+T_k}^{(k)}(W_t-\bar{W}^{(k)})\right] \\
&+ 2\mathbb{E}\left[(W_t-\bar{W}^{(k)})^\top P_{1+T_k}^{(k)}\cdots P_t^{(k)}H_f^{(k)}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)\right] \\
&+ \mathbb{E}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)^\top H_f^{(k)}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)
\end{aligned}
\tag{36}
$$

Since $\mathbb{E}[n_t]=0, \mathbb{E}\left[(W_t-\bar{W}^{(k)})^\top P_{1+T_k}^{(k)}\cdots P_t^{(k)}H_f^{(k)}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)\right] = 0$. Meanwhile, we notice the data samplings between different iteration are independent, thus $\mathbb{E}\left[(n_i)^\top n_j\right] = 0 \quad (i \neq j)$. We obtain

$$
\begin{aligned}
&\mathbb{E}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)^\top H_f^{(k)}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right) \\
&= \mathbb{E}\left(\sum_{i,j=1+T_k}^{t} \eta_i n_i^\top P_{i+1}^{(k)}\cdots P_t^{(k)}H_f^{(k)}P_t^{(k)}\cdots P_{j+1}^{(k)}\eta_j n_j\right) \\
&= \sum_{i=1+T_k}^{t} \mathbb{E}\left(\eta_i n_i^\top P_{i+1}^{(k)}\cdots P_t^{(k)}H_f^{(k)}P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)
\end{aligned}
\tag{37}
$$

Therefore, Eq.(36) can be reformulated as follows,

$$
\begin{aligned}
&\mathbb{E}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)^\top H_f^{(k)}\left(\sum_{i=1+T_k}^{t} P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right) \\
&= \mathbb{E}\left[(W_t-\bar{W}^{(k)})^\top P_{1+T_k}^{(k)}\cdots P_t^{(k)}H_f^{(k)}P_t^{(k)}\cdots P_{1+T_k}^{(k)}(W_t-\bar{W}^{(k)})\right] \\
&+ \sum_{i=1+T_k}^{t} \mathbb{E}\left(\eta_i n_i^\top P_{i+1}^{(k)}\cdots P_t^{(k)}H_f^{(k)}P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right) \\
&:= B + V
\end{aligned}
\tag{38}
$$

where we denote $B := \mathbb{E}\left[(W_t-\bar{W}^{(k)})^\top P_{1+T_k}^{(k)}\cdots P_t^{(k)}H_f^{(k)}P_t^{(k)}\cdots P_{1+T_k}^{(k)}(W_t-\bar{W}^{(k)})\right]$ bias term and $V := \sum_{i=1+T_k}^{t} \mathbb{E}\left(\eta_i n_i^\top P_{i+1}^{(k)}\cdots P_t^{(k)}H_f^{(k)}P_t^{(k)}\cdots P_{i+1}^{(k)}\eta_i n_i\right)$ variance term.

Denote $u_i^{(k)} \quad i \in \{1, 2, \cdots, N\}$ are linearly independent eigenvectors, which can form the basis for the N dimension vector space. Then we have $H_f^{(k)}(\xi)u_i^{(k)} = \lambda_i^{(k)}u_i^{(k)} \quad (i = 1, 2, \cdots, N)$ and the initial deviation from the optimal solution can be expressed as $W_t - \bar{W}^{(k)} = \sum_{i=1}^{N} s_i^{(k)}u_i^{(k)}$ where $s_i^{(k)}$ are the coefficients corresponding to the eigenvector components. Combined with Lemma 1, we obtain

Case 1 ($\varphi > 2$)**:**

$$
\begin{aligned}
B &= (W_t - \bar{W}^{(k)})^\top P_{1+T_k}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{1+T_k}^{(k)} (W_t - \bar{W}^{(k)}) \\
&= \Big[(I - \eta_t H_f^{(k)}) \cdots (I - \eta_{1+T_k} H_f^{(k)})(W_{1+T_k} - \bar{W}^{(k)})\Big]^\top H_f^{(k)} (I - \eta_t H_f^{(k)}) \cdots (I - \eta_{1+T_k} H_f^{(k)})(W_{1+T_k} - \bar{W}^{(k)}) \\
&= \sum_{i=1}^{N} (s_i^{(k)})^2 \|u_i^{(k)}\|_2^2 \lambda_i^{(k)} \prod_{j=1+T_k}^{t} \Big[(1 - \eta_j \lambda_i^{(k)})\Big]^2 \\
&\le \sum_{i=1}^{N} (s_i^{(k)})^2 \|u_i^{(k)}\|_2^2 \lambda_i^{(k)} \exp\Big(-2\lambda_i^{(k)} \eta_{\min}(t - T_k)\Big) \\
&\quad \cdot \Big(\frac{4(T_{k+1} - T_k) + (\varphi - 2)\pi}{4(T_{k+1} - T_k) + (\varphi - 2)\pi(t - T_k)}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{(2(t - T_k) - 1)\pi}{2(T_{k+1} - T_k)}\right)\right)(T_{k+1} - T_k)}{(\varphi - 2)\pi}} \\
&\le \Big(\frac{4(T_{k+1} - T_k) + (\varphi - 2)\pi}{4(T_{k+1} - T_k) + (\varphi - 2)\pi(t - T_k)}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{(2(t - T_k) - 1)\pi}{2(T_{k+1} - T_k)}\right)\right)(T_{k+1} - T_k)}{(\varphi - 2)\pi}} \\
&\quad \cdot \exp\Big(-2\lambda_i^{(k)} \eta_{\min}(t - T_k)\Big) \lambda_u^{(k)} \|W_{1+T_k} - \bar{W}^{(k)}\|_2^2
\end{aligned}
\tag{39}
$$

In addition,

$$
\begin{aligned}
V &= \sum_{i=_{1+T_k}}^{t} \mathbb{E}\left(\eta_i n_i^\top P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} \eta_i n_i\right) \\
&= \sum_{i=_{1+T_k}}^{t} \eta_i^2 \mathbb{E}\left[\mathrm{tr}\left(n_i^\top P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} n_i\right)\right] \\
&= \sum_{i=_{1+T_k}}^{t} \eta_i^2 \mathbb{E}\left[\mathrm{tr}\left(P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} n_i n_i^\top\right)\right] \\
&= \sum_{i=_{1+T_k}}^{t} \eta_i^2 \mathrm{tr}\left[P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} \mathbb{E}\left(n_i n_i^\top\right)\right] \\
&\le \sigma^2 \sum_{i=_{1+T_k}}^{t} \eta_i^2 \mathrm{tr}\left[P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} H_f^{(k)}\right] \\
&= \sigma^2 \sum_{i=_{1+T_k}}^{t} \eta_i^2 \sum_{j=1}^{N} (\lambda_j^{(k)})^2 \prod_{q=i+1}^{t} \left(1 - \eta_q \lambda_j^{(k)}\right)^2 \\
&\le \sigma^2 \sum_{i=_{1+T_k}}^{t} \eta_i^2 \sum_{j=1}^{N} (\lambda_j^{(k)})^2 \exp\Big(-2\lambda_j^{(k)} \eta_{\min}(t - i)\Big) \\
&\quad \cdot \Big(\frac{4(T_{k+1} - T_k) + (\varphi - 2)\pi(i - T_k)}{4(T_{k+1} - T_k) + (\varphi - 2)\pi(t - T_k)}\Big)^{\frac{4\lambda_l^{(k)}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{(2(t - T_k) - 1)\pi}{2(T_{k+1} - T_k)}\right)\right)(T_{k+1} - T_k)}{(\varphi - 2)\pi}}
\end{aligned}
\tag{40}
$$

where the second equality comes from cyclic property of trace and the first inequality comes from $\mathbb{E}_\xi\left[n_t n_t^\top\right] \preceq \sigma^2 H_f^{(k)}$.

Case 2 ($\varphi < 2$):

$$
\begin{aligned}
B &= (W_t - \bar{W}^{(k)})^\top P_{1+T_k}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{1+T_k}^{(k)} (W_t - \bar{W}^{(k)}) \\
&= \left[(I - \eta_t H_f^{(k)}) \cdots (I - \eta_{1+T_k} H_f^{(k)})(W_{1+T_k} - \bar{W}^{(k)})\right]^\top H_f^{(k)} (I - \eta_t H_f^{(k)}) \cdots (I - \eta_{1+T_k} H_f^{(k)})(W_{1+T_k} - \bar{W}^{(k)}) \\
&= \sum_{i=1}^{N} (s_i^{(k)})^2 \|u_i^{(k)}\|_2^2 \lambda_i^{(k)} \prod_{j=1+T_k}^{t} \left[(1 - \eta_j \lambda_i^{(k)})\right]^2 \\
&\leq \sum_{i=1}^{N} (s_i^{(k)})^2 \|u_i^{(k)}\|_2^2 \lambda_i^{(k)} \exp\Big(-2\lambda_i^{(k)} \eta_{\min}(t - T_k)\Big) \\
&\quad \cdot \Big(\frac{(2\varphi + 2\pi - \varphi\pi) - \frac{(2-\varphi)\pi}{(T_{k+1} - T_k)}(t - T_k - 0.5)}{(2\varphi + 2\pi - \varphi\pi) + \frac{(2-\varphi)\pi}{2(T_{k+1} - T_k)}}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{(2(t - T_k) - 1)\pi}{2(T_{k+1} - T_k)}\right)\right)(T_{k+1} - T_k)}{(2-\varphi)\pi}} \\
&\leq \Big(\frac{(2\varphi + 2\pi - \varphi\pi) - \frac{(2-\varphi)\pi}{(T_{k+1} - T_k)}(t - T_k - 0.5)}{(2\varphi + 2\pi - \varphi\pi) + \frac{(2-\varphi)\pi}{2(T_{k+1} - T_k)}}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{(2(t - T_k) - 1)\pi}{2(T_{k+1} - T_k)}\right)\right)(T_{k+1} - T_k)}{(2-\varphi)\pi}} \\
&\quad \cdot \exp\Big(-2\lambda_i^{(k)} \eta_{\min}(t - T_k)\Big) \lambda_u^{(k)} \|W_{1+T_k} - \bar{W}^{(k)}\|_2^2
\end{aligned}
\tag{41}
$$

In addition,

$$
\begin{aligned}
V &= \sum_{i=1+T_k}^{t} \mathbb{E}\left(\eta_i n_i^\top P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} \eta_i n_i\right) \\
&= \sum_{i=1+T_k}^{t} \eta_i^2 \mathbb{E}\left[\mathrm{tr}\left(n_i^\top P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} n_i\right)\right] \\
&= \sum_{i=1+T_k}^{t} \eta_i^2 \mathbb{E}\left[\mathrm{tr}\left(P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} n_i n_i^\top\right)\right] \\
&= \sum_{i=1+T_k}^{t} \eta_i^2 \mathrm{tr}\left[P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} \mathbb{E}\left(n_i n_i^\top\right)\right] \\
&\leq \sigma^2 \sum_{i=1+T_k}^{t} \eta_i^2 \mathrm{tr}\left[P_{i+1}^{(k)} \cdots P_t^{(k)} H_f^{(k)} P_t^{(k)} \cdots P_{i+1}^{(k)} H_f^{(k)}\right] \\
&= \sigma^2 \sum_{i=1+T_k}^{t} \eta_i^2 \sum_{j=1}^{N} (\lambda_j^{(k)})^2 \prod_{q=i+1}^{t} \left(1 - \eta_q \lambda_j^{(k)}\right)^2 \\
&\leq \sigma^2 \sum_{i=1+T_k}^{t} \eta_i^2 \sum_{j=1}^{N} (\lambda_j^{(k)})^2 \exp\Big(-2\lambda_j^{(k)} \eta_{\min}(t - i)\Big) \\
&\quad \cdot \Big(\frac{(2\varphi + 2\pi - \varphi\pi) - \frac{(2-\varphi)\pi}{(T_{k+1} - T_k)}(t - T_k - 0.5)}{(2\varphi + 2\pi - \varphi\pi) - \frac{(2-\varphi)\pi}{(T_{k+1} - T_k)}(i - T_k - 0.5)}\Big)^{\frac{4\lambda_i^{(k)}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{(2(t - T_k) - 1)\pi}{2(T_{k+1} - T_k)}\right)\right)(T_{k+1} - T_k)}{(2-\varphi)\pi}}
\end{aligned}
\tag{42}
$$

where the second equality comes from cyclic property of trace and the first inequality comes from $\mathbb{E}_\xi\left[n_t n_t^\top\right] \preceq \sigma^2 H_f^{(k)}$.

□

In this proof, we employ assumption $\mathbb{E}_{\xi}\left[n_t n_t^{\top}\right] \preceq \sigma^2 H_f^{(k)}$ used in the work [16, 36]. Moreover, we acknowledge that the idea of the proof is inspired by the approach introduced in [16, 36]

E Experimental details

E.1 Baselines

We detail the baselines as follows,

- **Step schedule(SS)** reduces the learning rate in a piecewise manner. It implements discrete learning rate drops at predefined epoch intervals following schedule $g(t) = d_{\lfloor t/t_s \rfloor}$, where t_s is the predefined decaying step and $d_i (d_i \geq d_{i+1})$ is predefined decaying scalar [16]. A typical implementation decreases the learning rate by a factor of 0.1 after 50% of the epochs and further by a factor of 0.01 after 75% of the epochs [21]. However, it requires careful tuning of step timing since abrupt changes may destabilize optimization. In this paper, we employ such setting of step schedule ($\times$ 0.1 after 50% and $\times$ 0.01 after 75%) for all our experiments.
- **Cosine schedule(CS)** controls the t-th iteration learning rate η_t following the mathematical formulation as $\eta_t = \eta_{min} + \frac{1}{2}(\eta_0 - \eta_{min})(1 + \cos(\frac{t\pi}{T}))$ where T is total iterations. It provides smooth transition between learning rates, shown to improve final model accuracy step schedule. Moreover, it is proposed to mitigate abrupt decreases in the learning rate that could hinder models from escaping local minima. This approach has demonstrated effectiveness, particularly in transformer training [33]. Besides, it includes restart mechanisms where T becomes the cyclic period.
- **Cyclical Learning Rates (CLR)** oscillates between base learning rate η_{min} and maximal learning rate η_{max} with triangular and triangular2 policies. The cyclic property help traverse loss landscape barrier, so that it enables neural networks to train significantly faster, often by an order of magnitude, compared to standard training methods [43, 44]. Besides, the commonly-used OneCycle learning rate schedule can be seen as the one period version of Cyclical Learning Rates. In our experiments, this baseline **CLR** includes Cyclical and OneCycle schedules(**1C**). For CLR, we set linear mode for each period and period is two.
- **Budgeted training(BT)** is conducted by the proposed budget-aware setting of existing learning rate schedule under epoch budget [30]. It reformulates standard schedules such as linear, step and cosine schedule as functions of remaining budget. The classic linear decay version is formulated as $\eta_t = \eta_0 \times (1 - t/T)$ where T is total iterations. It is particularly effective for hyper-parameter tuning.
- **Reflected Exponential (REX)** controls the t-th iteration learning rate η_t following the mathematical formulation as $\eta_t = \eta_0 \left(\frac{1 - t/T}{0.5 + 0.5 \times (1 - t/T)} \right)$. It is inspired by the performance of delayed linear schedules, aggressively reducing the learning rate near the end of training and reflecting an exponential decay [5].

E.2 Evaluation benchmarks of language task

We evaluate models on a diverse set of open benchmarks. The benchmarks are listed in Table 5. For language models evaluation, we adopt the [15] for standardized performance comparisons.

E.3 Implementation Details

Consistent with CIFAR, ImageNet and OLMo practices, we use step-wise scheduling for all training procedures.

Vision tasks on CIFAR CIFAR10/100 are simple benchmark datasets that are widely used for quick and efficient evaluation of deep learning tasks. CIFAR10 consists of a training split of 50000 examples and a test split of 10000 examples. Each example is a 32×32 color image in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). CIFAR100 comprises 60000 32×32 pixels color images which are divided into 100 classes. Each class contains 600 images and the classes are grouped into 20 super-classes. Each image comes with a 'fine' label (the class to which it belongs) and a 'coarse' label (the super-class to which it belongs). We use the `torch.optim.SGD` and `torch.optim.AdamW` API to configure the optimizer. For vision tasks in Section 4.1, we all adopt the SGD optimizer. For ablation studies in Section 4.3, we adopt the AdamW optimizer. We

Table 5: Evaluation benchmarks of language task

Datasets name	abbreviation
ARC-Easy [10]	ARC-E
ARC-Challenge [10]	ARC-C
HellaSwag [55]	HSWAG
PIQA [3]	PIQA
WinoGrande [39]	WG
OpenbookQA [35]	OBQA
BoolQ [9]	BoolQ
SciQ [50]	SciQ
COPA [18]	COPA
CommonsenseQA [46]	CSQA
SocialIQA [40]	SIQA
MMLU stem [22]	STEM
MMLU humanities [22]	HUM
MMLU social sci [22]	SOC
MMLU other [22]	OTH

initialize the learning rate to 1e-1, set the batch size to 128. Each epoch consists of 50000/128 + 1 = 391 steps. We leave the weight decay value as 5e-4, and complete the training task on PyTorch 2.4.1+cu118 with RTX 5880 and RTX 3090.

Vision tasks on ImageNet To evaluate the scalability of our schedule on larger-scale dataset, we conduct experiments on the ImageNet dataset, a dataset that more closely reflects real-world visual recognition challenges. ImageNet consists of approximately 1.28 million training images and 50000 validation images across 1000 classes, which also come with an official dataset split. This benchmark, particularly with the ResNet-50 architecture, has been extensively studied and is widely regarded as a standard evaluation setup. We adopt the common ResNet50 architecture, and separately train it with different learning rate schedulers. To provide a comprehensive evaluation, we conduct experiments using both the classical SGD optimizer and the popularized AdamW optimizer. For the experiments with SGD, we set the peak learning rate to 0.5 and apply the weight decay factor of 1e-4. For the experiments using AdamW, the peak learning rate is set to 3e-3, with a decoupled weight decay of 0.02. Following default configuration which requires learning rate warmup, we integrate 10% warmup phases of the total training tokens into all schedules. For schedules that inherently include a warmup-like behavior, such as the 1C schedule, where the learning rate initially rises from zero to a target value, we retain their original settings during the ascending phase. We set the batch size to 2048 and complete these training tasks with 8 NVIDIA A100 GPUs.

Language tasks For language tasks, we use the OLMo model. OLMo [19] is a decoder-only transformer-based language model that builds upon the vanilla Transformer architecture [49]. It incorporates several key improvements inspired by recent large language models (LLMs) such as PaLM [8], the LLaMA family [48], OpenLM and Falcon [1]. Notably, OLMo stands out as a truly open language model, offering full transparency through its open training data, released training/evaluation code, intermediate model checkpoints, and comprehensive training logs, making it a valuable resource for reproducibility and further research.

For the training hyper-parameters on OLMo, we primarily adopt the configuration outlined in OLMo [19]. We conduct experiment on H100 and A100. Following default configuration which requires learning rate warmup, we integrate 10% warmup phases of the total training tokens into all schedules. For schedules that inherently include a warmup-like behavior, such as the 1C schedule, where the learning rate initially rises from zero to a target value, we retain their original settings during the ascending phase. Besides, large language models commonly provide multiple scales to accommodate different compute constraints. We adjust both model size and training steps to explore budgets. To ensure robust conclusions, we maintain a data-to-parameters ratio (D/N) of 500 across scales. The specific configurations are as follows, 36M-parameter model is trained on 50 billion tokens, 73M-parameter model is trained on 50 billion tokens, 151M-parameter model is trained on 75 billion tokens, 300M-parameter model is trained on 150 billion tokens. This scaling strategy allows us

to systematically study the relationship between model size, training data, and performance while controlling for the D/N ratio. The OLMo model is trained using AdamW optimizer. The detailed configuration are shown in Table 6.

Table 6: Configurations of OLMo

Model scale	36M	73M	151M	300M
token number	50B	50B	75B	150B
Layers (L)	12	16	20	24
Hidden dim (D)	512	640	768	1024
Attention heads	8	10	8	8
Inner dim (H)	1280	1536	2240	2688
Vocab size (V)	100,352			

Our implementation strictly follows the PyTorch official API, i.e. it is inherited from the base class

`torch.optim.lr_scheduler.LRScheduler`

. It maintains full compatibility with all

`torch.optim`

optimizers. The complete production-ready code is publicly available.

The code of UBA schedule is available at `https://github.com/Ttt-answer/UBA.git`.

E.4 Overall experiment results for Vision Classification Tasks

Overall results for the vision classification tasks are presented in Table 7, depicting validation accuracy on vision benchmarks (e.g., CIFAR10/100 and ImageNet) using different architectures (ResNet18, ResNet34, ResNet50). We conduct several independent runs for error bars. Due to computational constraints, we use two independent experimental results for ImageNet-ResNet50. It is obvious that our proposed UBA schedule outperform baselines, achieving the highest validation accuracy over both small-scale and large-scale benchmarks across all training budgets.

Our analysis reveals the following key findings. (i) On CIFAR10-ResNet18 and CIFAR100-ResNet34 task, UBA shows the strongest performance across all training budgets. On the large-scale ImageNet benchmark with ResNet50, UBA maintains consistent superiority. The consistent improvements on both small-scale and large-scale benchmarks highlight UBA's **generalizability across scales and data regimes**. (ii) UBA outperforms baselines not only at 100% training budget but also at 25% and 50% iteration budgets, demonstrating its **budget efficiency**, i.e.,effectiveness in computation-constrained scenarios. (iii)Notably, UBA shows robust performance even while the second-best schedule varies depending on both the datasets, architectures and training budgets. This instability among competing schedules suggests their performance is highly sensitive to datasets-architectures characteristics and training dynamics at different learning phases. Therefore, for practitioners seeking reliable schedules without extensive method selection, UBA provides a default-strong choice. UBA eliminates the need for case-by-case baseline comparison and delivers **stable superiority and reliability** regardless of datasets, architectures, training budgets and scales.

E.5 Overall experiment results for language models

For most tasks, we conduct a comprehensive evaluation across six representative baselines to ensure broad coverage. For large model with 300M parameters, we focus our comparison on the top three performing baselines due to the computational tractability.

The overall performance of the OLMo models is presented in Table 8. We also plot training curves in Figure 7,8, 9 and 10, showing training and validation losses along with downstream evaluation results across model sizes ranging from 36M to 300M parameters and training data scales from 50B to 150B tokens. As shown, UBA schedule consistently achieves lower training and validation losses while delivering superior downstream performance.

Table 7: Generalization accuracy.

Dataset	Network	Method	training budget (epoch(%))		
			75 (25%)	150(50%)	300(100%)
CIFAR10	ResNet18	SS	92.71±0.4243	93.45±0.4822	93.61±0.3035
		CS	94.21±0.5116	94.24±0.2419	95.52±0.2433
		CLR	92.13±0.2610	92.99±0.5459	94.92±0.3811
		1C	94.23±0.1838	95.03±0.1353	95.35±0.2984
		BT	94.18±0.3546	94.79±0.4605	95.68±0.1168
		REX	94.49±0.3225	95.10±0.1929	95.45±0.1350
		UBA(ours)	**94.57±0.3281**	**95.26±0.2150**	**95.65±0.1589**
			75 (25%)	150(50%)	300(100%)
CIFAR100	ResNet34	SS	71.09±0.2676	75.34±0.1802	77.24±0.1808
		CS	64.12±0.1808	69.84±0.7580	74.29±0.5958
		CLR	73.33±0.5635	73.41±0.2835	74.74±0.2266
		1C	73.38±0.3227	75.04±0.6382	77.86±0.2239
		BT	72.75±0.2611	75.35±0.2423	78.58±0.2705
		REX	73.64 ±0.5635	75.14±0.3110	78.04±0.1430
		UBA(ours)	**74.60±0.1508**	**76.63±0.2678**	**78.95±0.2818**
			75 (25%)	150(50%)	300(100%)
ImageNet	ResNet50	SS	74.74±0.1386	77.24±0.3861	78.56±0.6435
		CS	75.84±0.2220	77.95±0.2164	79.07±0.0438
		CLR	73.90±1.4609	76.04±1.6419	77.84±1.3053
		1C	75.83±0.7792	77.74±0.5218	78.90±0.2956
		BT	75.90 ±0.2786	77.81±0.4624	78.86± 0.2814
		REX	75.85±0.8047	77.66 ±0.8853	78.66 ±0.2828
		UBA(ours)	**76.29±0.4031**	**78.26± 0.1640**	**79.39±0.0170**

From Table 8, UBA achieves state-of-the-art performance on approximately 50% of the benchmarks across all scales while the second-best schedule varies. Furthermore, it achieves *consistent superior average performance* among all baselines. It highlights the stable superiority and reliability of UBA, providing a *default-strong choice*. Moreover, it demonstrates significant improvements in SciQ-73M($+1.7$) and ARC-E-300M($+2.63$), highlighting its ability to enhance generalization across diverse benchmarks. Besides, in Figure 2, UBA consistently achieves lower training loss and validation loss throughout the training, indicating the *efficient training ability* and downstream performance enhancement. Notably, while task-specific tuning of φ can further improve model performance, we intentionally fix its value across all tasks and architectures to ensure fair evaluation. Remarkably, even with this universal φ setting, UBA consistently outperforms baselines on diverse benchmarks, demonstrating *inherent robustness* to varying benchmarks and model scales.

E.6 Performance across different optimizers

Our scheduling strategy originates from the solution to a optimization problem under gradient descent dynamics. While modern optimizer (e.g., AdamW [32]) introduce additional momentum and adaptive mechanisms, they maintain the fundamental property of performing gradient-based updates. To verify the performance of schedule, we conduct cross-optimizer ablation studies. We evaluate the model on CIFAR100 and ImageNet datasets using AdamW optimizer. For AdamW, we use a learning rate of 0.003 and weight decay of 0.0001. We report the validation accuracy for each configuration. The results are summarized in Table 9.

As shown in Table 9, our UBA schedule achieves state-of-the-art validation accuracy on both SGD and AdamW optimizers, consistently outperforming baselines across CIFAR100-ResNet34 and ImageNet-ResNet50 benchmarks. This demonstrates that although our schedule is theoretically derived from standard gradient descent dynamics, it generalizes effectively to modern optimizers like AdamW — despite their additional momentum and adaptive mechanisms — highlighting its broad applicability.

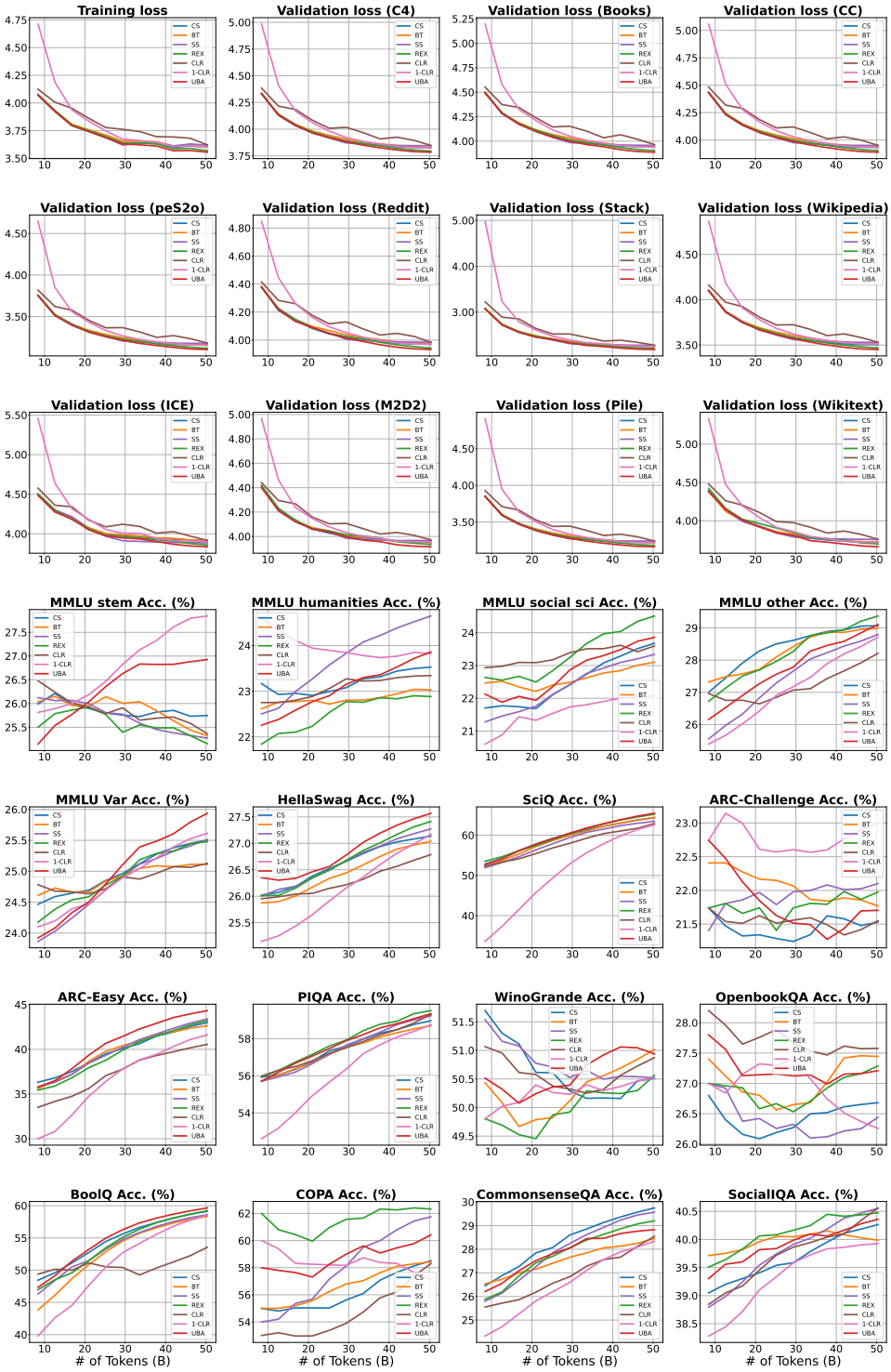


Figure 7: Overall performance for language tasks on OLMo-36M.

E.7 Parameter analysis of φ

In this subsection, we perform a sensitivity analysis of the key parameter φ in equation (5) , which controls the variation speed of learning rate. Our experiments systematically evaluate

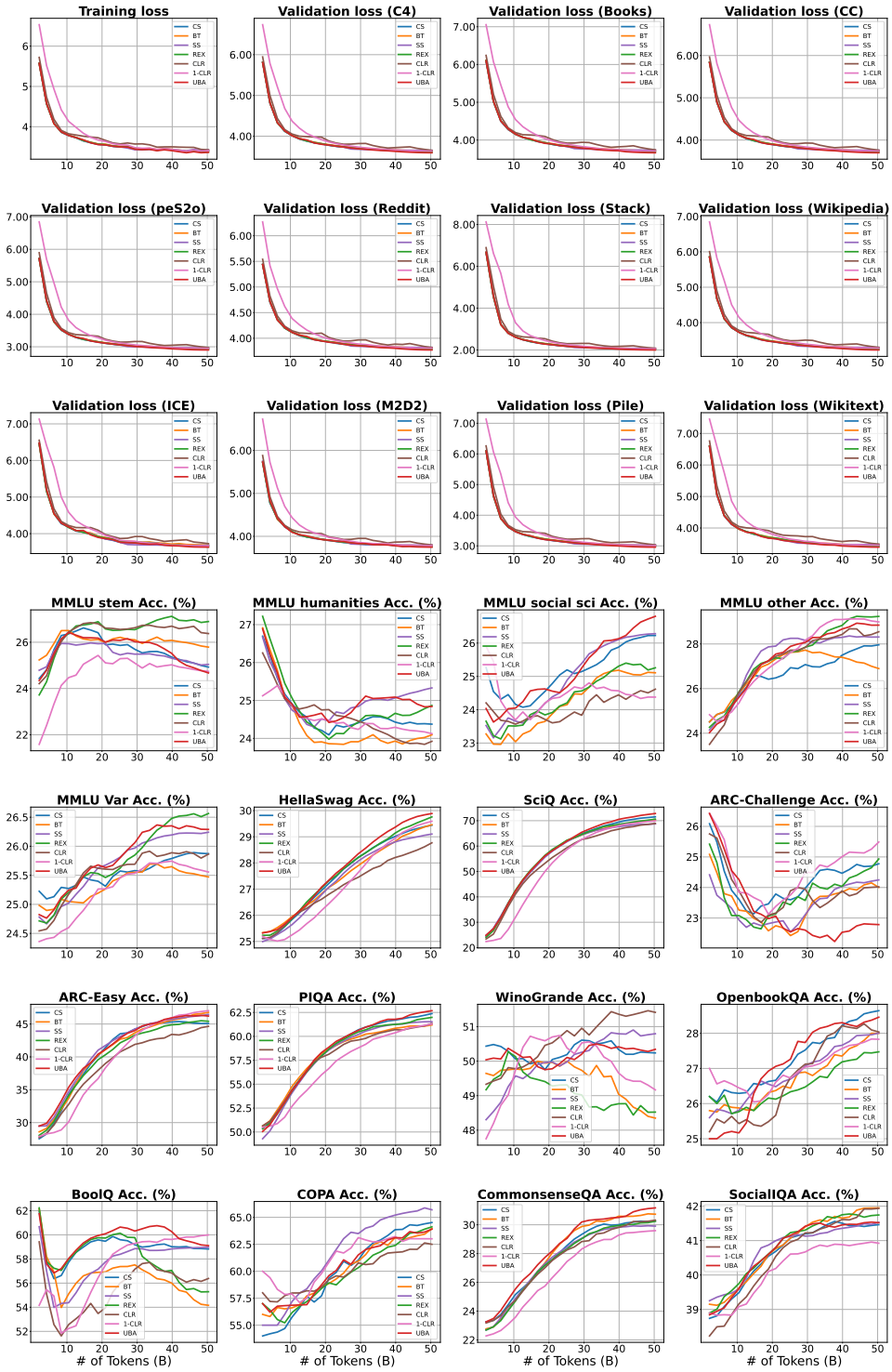


Figure 8: Overall performance for language tasks on OLMo-73M.

$\varphi \in \{0.25, 0.5, 1.0, 2.5, 5, 10\}$ across different optimization scenarios. The results are plotted in Figure 11 and listed in Table 10).

Observations From the experimental results, we find an interesting dichotomy of φ. Phenomenon(a): By choose proper φ, UBA demonstrates consistent performance gains over all baselines

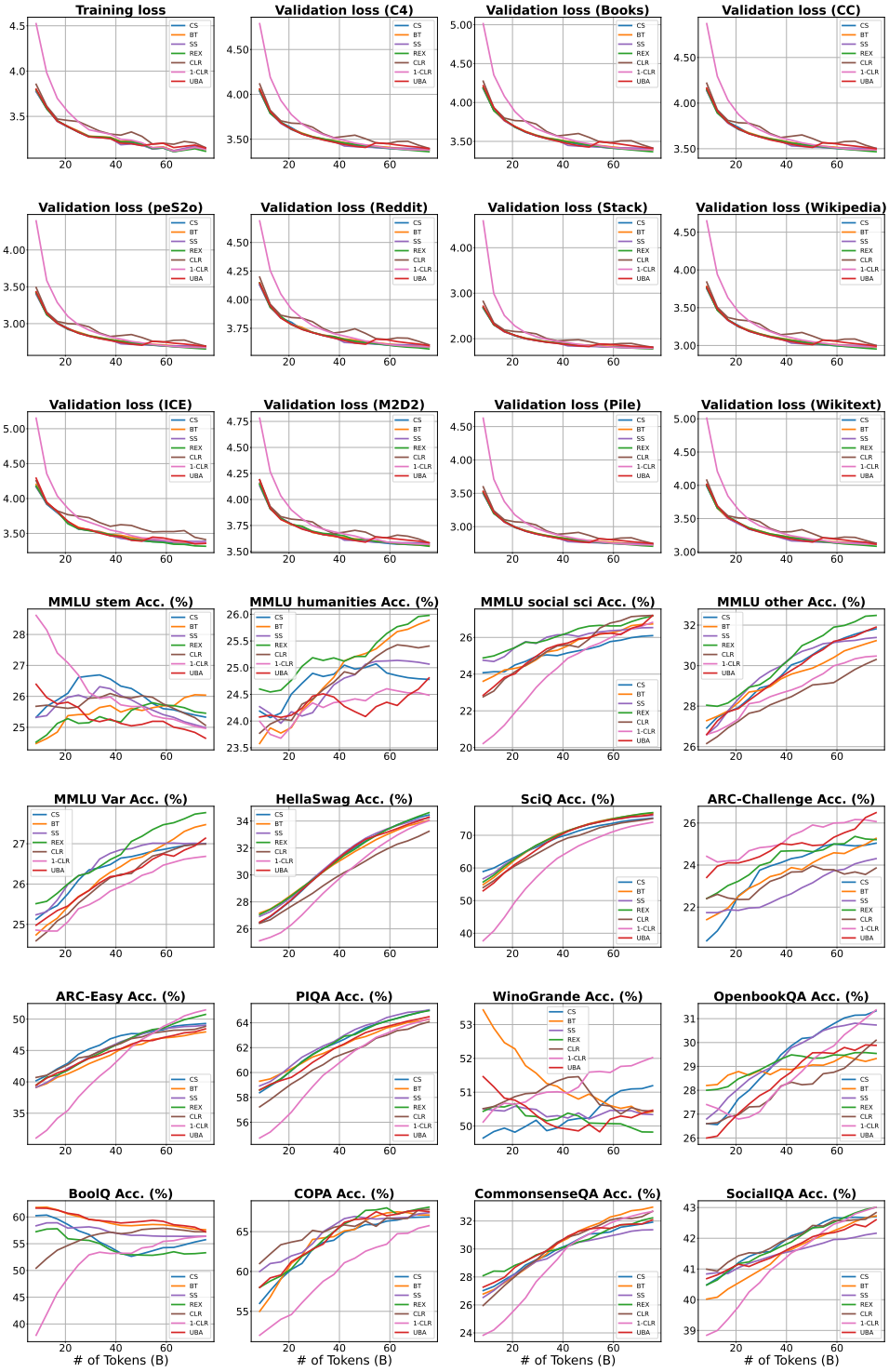


Figure 9: Overall performance for language tasks on OLMo-150M.

across the full spectrum of training budgets (25%, 50% and 100%), irrespective of optimizer choice (SGD/AdamW). This suggests the effectiveness of UBA is orthogonal to specific optimization strategies. Besides, with optimizer changed, UBA maintains comparable accuracy on the same dataset-architecture, which shows that UBA can exhaustively exploit the model's capacity regardless of optimizer variants. Phenomenon(b): As shown in Figure 11 and Table 10, on CIFAR100 dataset,

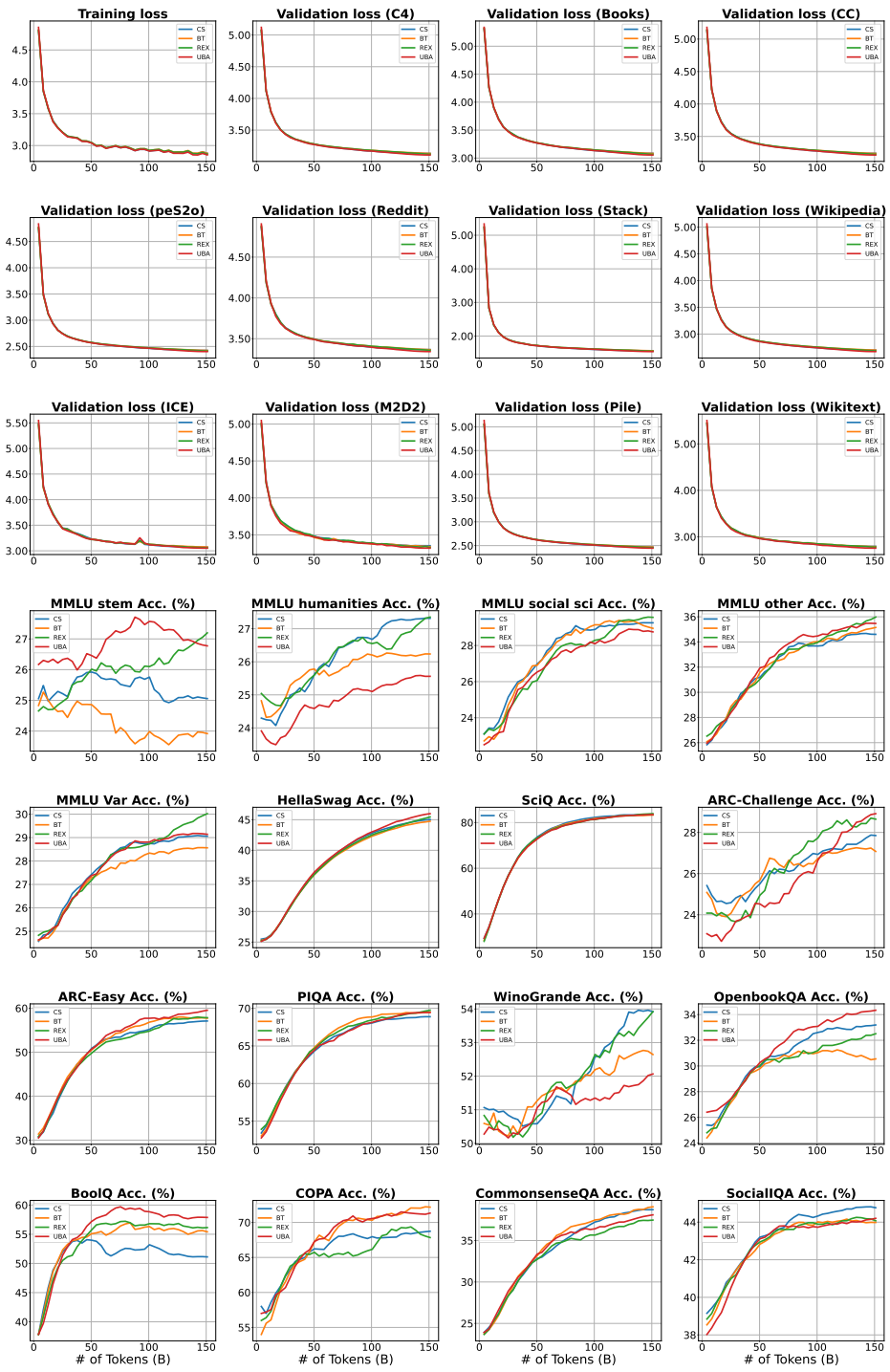


Figure 10: Overall performance for language tasks on OLMo-300M.

the best performance for AdamW is achieved with $\varphi = 1$, whereas SGD performs optimally with $\varphi = 2.5$. On ImageNet dataset, the best performance for AdamW is achieved with $\varphi = 0.5$, whereas SGD performs optimally with $\varphi = 10$. Our experiments reveal a notable trend: when using AdamW, a smaller φ yields better performance, whereas a larger φ is preferred for SGD.

Table 8: Performance comparison on OLMo model

Size	Sched.	Benchmark(accuracy %)										
		PIQA	HSWAG	OBQA	SciQ	ARC-E	ARC-C	COPA	SIQA	SOC	OTH	Avg.
36M	CS	59.74	27.36	26.80	66.50	44.74	21.74	60.00	40.58	24.33	29.09	40.09
	BT	59.41	27.33	27.40	67.20	43.68	21.40	59.00	39.82	23.48	29.13	39.79
	SS	60.12	27.63	27.20	65.30	45.26	22.41	63.00	40.89	23.87	29.62	40.53
	REX	60.17	27.82	**27.80**	68.10	45.09	22.41	62.00	40.63	**25.18**	30.02	40.92
	CLR	**61.15**	27.21	27.60	67.50	42.11	22.07	62.00	**41.45**	24.29	29.38	40.48
	1C	60.17	27.91	25.80	67.40	43.68	**23.41**	55.00	40.02	22.26	29.86	39.55
	UBA	60.39	**27.98**	27.40	**68.20**	**45.79**	21.74	**63.00**	40.69	24.33	**30.14**	**40.97**
73M	CS	63.06	29.68	28.80	72.60	45.09	25.08	65.00	41.56	26.24	28.17	42.53
	BT	61.70	29.79	28.00	71.40	47.54	23.41	**66.00**	**41.97**	25.05	26.48	42.13
	SS	61.53	29.30	28.20	69.40	47.19	24.41	65.00	41.76	26.32	28.33	42.14
	REX	62.46	**30.22**	27.60	72.20	45.09	26.09	65.00	41.81	25.52	**29.34**	42.53
	CLR	62.30	29.38	27.80	70.40	45.44	24.08	62.00	42.02	25.05	29.05	41.75
	1C	61.81	29.90	27.80	71.10	**47.54**	**26.42**	63.00	40.79	24.37	28.81	42.15
	UBA	**63.17**	30.09	**28.80**	**74.30**	45.79	22.74	65.00	41.45	**27.13**	28.85	**42.73**
150M	CS	**66.00**	35.19	32.00	76.60	49.82	25.42	67.00	42.84	26.24	32.27	45.34
	BT	64.85	34.95	29.80	78.20	48.95	26.42	67.00	42.99	26.83	31.87	45.19
	SS	65.45	34.76	30.60	77.10	49.65	24.75	68.00	42.37	26.58	31.59	45.09
	REX	65.56	**35.72**	29.40	78.30	52.46	25.08	**69.00**	43.30	27.68	32.64	45.91
	CLR	65.07	34.76	31.60	77.20	50.70	25.08	68.00	**43.71**	27.30	31.23	45.47
	1C	65.29	35.16	**32.80**	76.30	**53.16**	25.75	67.00	43.45	27.64	30.62	45.72
	UBA	65.23	35.50	29.80	**78.30**	50.35	**27.42**	67.00	43.50	**29.29**	**32.72**	**45.91**
300M	CS	68.88	45.32	33.40	83.20	57.19	27.76	69.00	**44.58**	29.25	34.57	49.32
	BT	69.64	45.21	30.80	83.70	57.37	26.42	72.00	43.96	28.66	35.49	49.33
	REX	**70.18**	46.30	33.00	**84.40**	57.72	28.43	67.00	43.71	**29.50**	**36.74**	49.70
	UBA	69.48	**46.44**	**34.60**	83.90	**60.35**	**29.10**	**72.00**	44.42	28.53	35.41	**50.42**

Analysis Furthermore, we intend to explain the dual behavior of φ through theoretical analyses (Proposition 1 and Theorem 1). We attribute Phenomenon (a) to UBA's optimal scheduling dynamics, governed by the φ value. Moreover, Phenomenon (b) reveals a fundamental dichotomy in φ value selection between optimizers. We attribute this phenomenon(b) to the preconditioning effect of AdamW. Unlike SGD, AdamW adapts the learning rate by scaling gradients with the square root of the second moment estimate $\sqrt{v_t}$. In regions where gradients g_t are large, the surrounding landscapes are steep. The denominator $\sqrt{v_t}$ is also large, effectively reducing the learning rate. This adaptive behavior mimics preconditioning, implicitly lowering the condition number of the optimization landscape.

Our theoretical framework (Proposition 1) establishes a connection between parameter φ and the condition number of the landscape around local minima: smaller φ is favored for well-conditioned problems (low condition number), while larger φ is beneficial for large condition number scenarios. Since preconditioning effect of AdamW naturally mitigates ill-conditioning, the schedule with smaller φ is preferred. In contrast, SGD lacks such adaptive mechanisms, thus a larger φ is more suitable for this situation. This alignment between theory and experiment underscores the importance of tailoring φ to the optimizer's characteristics. *Thus, we recommend selecting higher values of φ when facing challenging optimization difficulty, while employing lower φ values in scenarios with smoother optimization difficulty.*

E.8 Performance across different initial learning rates

Since each schedule will have a different optimal learning rate in practice, to thoroughly address this concern, we conducted additional experiments to evaluate the performance of all schedules with different learning rates.

Table 9: Cross-optimizer ablation studies. We list validation accuracy for vision classification tasks across SGD and AdamW.

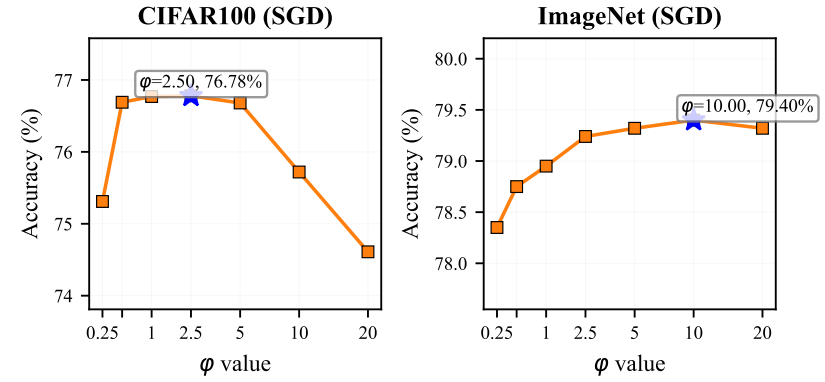
Dataset -network	Optimizer	Schedule	training budget (epoch(%))		
			30 (25%)	60(50%)	120(100%)
CIFAR100 -ResNet34	SGD	SS	70.85	*75.58*	77.28
		CS	63.88	68.84	70.43
		CLR	72.63	73.81	74.80
		1C	*73.72*	75.53	77.90
		BT	72.53	75.40	78.49
		REX	73.12	75.48	*77.99*
		UBA(ours)$\varphi = 5$	**74.57**	**76.68**	**78.97**
	AdamW	SS	64.01	70.93	73.65
		CS	62.56	68.91	72.98
		CLR	62.87	70.13	73.22
		1C	64.27	71.60	74.06
		BT	64.01	71.44	*74.34*
		REX	**65.19**	*71.72*	74.08
		UBA(ours)$\varphi = 0.5$	*64.44*	**72.10**	**74.48**
			75 (25%)	150(50%)	300(100%)
ImageNet -ResNet50	SGD	SS	74.84	76.96	78.10
		CS	*75.99*	*77.79*	*79.10*
		CLR	72.86	74.88	76.91
		1C	75.28	77.37	78.79
		BT	75.71	77.48	78.66
		REX	75.28	77.03	78.46
		UBA(ours)$\varphi = 5$	**76.00**	**77.99**	**79.32**
	AdamW	SS	74.64	77.51	79.01
		CS	75.68	78.10	79.04
		CLR	74.93	77.20	78.76
		1C	76.38	78.11	79.21
		BT	76.10	78.14	*79.05*
		REX	*76.42*	*78.28*	78.86
		UBA(ours) $\varphi = 0.5$	**76.57**	**78.37**	**79.38**

Table 10: Cross-φ ablation studies.

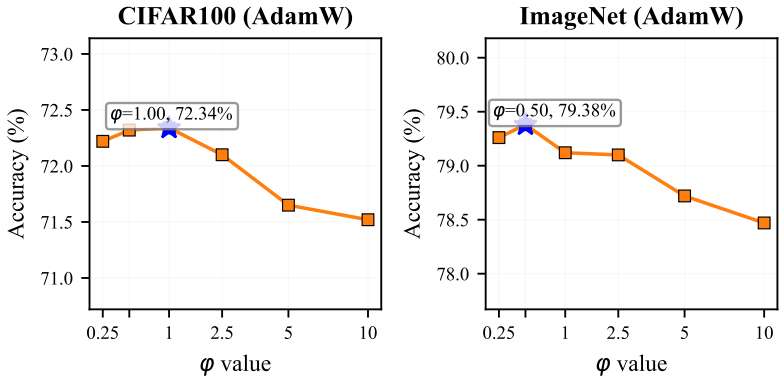
Dataset	Network	Epoch	Optimizer	UBA					
				φ=0.25	φ=0.5	φ=1	φ=2.5	φ=5	φ=10
CIFAR100	ResNet34	60	SGD	75.10	75.89	76.31	76.40	**76.68**	75.64
			AdamW	**72.10**	71.34	71.54	71.21	70.40	69.01
ImageNet	ResNet50	300	SGD	78.35	78.75	78.95	79.24	79.32	**79.40**
			AdamW	79.26	**79.38**	79.12	79.10	78.72	78.47

We conducted auxiliary experiments on CIFAR100-ResNet34 for 3 runs. For computational efficiency, these experiments were run for slightly fewer epochs than the main experiments, with results reported over three independent runs. Through current observation and search, we identified 0.1 as the proper initial learning rate for most schedules. We use 0.01 and 0.001 to confirm this choice maximizes validation accuracy across methods. Additional smaller or larger values such as 0.3 or 0.0001 were explored in preliminary studies, showing consistent trends but lower peak accuracy. Thus we do not choose them for comparison.

The experimental results are listed in the following table 11. We report results of different base learning rate and compare the performance with optimal learning rate of each schedule.



(a)



(b)

Figure 11: Visualization of cross-φ performance.

Table 11: Performance across different initial learning rates.

Dataset	Network	Schedule	Validation Accuracy		
			lr=0.1	lr=0.01	lr=0.001
CIFAR100	ResNet34	SS	77.28±0.2401	73.48±0.4267	67.98±0.1473
		CS	74.11±0.7690	74.66±0.5888	69.18±0.3736
		CLR	74.77±0.1861	74.26±0.3523	68.84±0.4613
		1C	77.73±0.1868	75.80±0.1701	69.20±0.2152
		BT	78.53±0.0751	77.11±0.1026	68.84±0.4613
		REX	78.11±0.1124	77.40±0.4477	69.88±0.1253
		UBA(ours)	79.02±0.2838	77.59±0.3522	70.50±0.0862

E.9 Performance across different periods

Our schedule has a periodic phase-based learning rate adjustment setting, where the learning rate at the k-th phase is dynamically determined by the k-th local minimum of the loss landscape. In the original conception, this design captures the optimization dynamics around successive local minima, with the hyper-parameter φ related to the difficulty of each phase. Specifically, φ controls the trade-off between exploration (large steps) and exploitation (small steps) within each phase.

To validate our scheduling strategy, we conduct experiments by varying K (see detailed results in Appendix E.9). We observe that when roughly setting φ as a constant in each phase, performance degrades as training progresses. This suggests that a fixed φ strategy cannot adapt to the changing

landscape of multi-phase scheduling. Since the optimization difficulty should decrease during training process, we drop φ as phase increases. By finely evaluate φ in each phase, it achieves performance improvement, confirming the need for dynamic control. It makes sense since multi-phase captures the dynamic features of the loss surface more finely, it needs careful selection for φ, where φ reflects optimization difficulty. These results highlight a fundamental trade-off: when φ values per phase for multi-phase scheduling can be finely evaluated, multi-phase scheduling better captures the loss landscape's non-stationary behavior than single-phase scheduling. However, selecting optimal φ values per phase for multi-phase remains non-trivial, which motivates future work on automated landscape-aware φ tuning.

Table 12: Performance on multi-phase version UBA.

Dataset	Network	φ	Original	Multi-phase		
				K=4	K=6	K=8
CIFAR100	ResNet34	5	76.68	76.54	75.37	75.05
		5×0.8^k		76.31	76.60	76.45

F Related work

Budgeted training Researchers face significant challenges in achieving optimal model performance under fixed hardware and limited time. To address this, budgeted training has emerged as a broad research domain focusing on optimizing performance while minimizing resource usage. This research spans multiple aspects, including computation efficiency, model compression, training stability, convergence improvement, and optimization [41]. By strategically integrating efficient training approaches, budgeted training advances cost-effective model development, offering pathways to achieve high performance under realistic constraints.

Budgeted training can be divided from three main perspectives: data, model, and optimization. From the data perspective, budgeted training emphasizes the allocation of resources between the dataset and the algorithms that process it, such as the balance between the model size and the amount of data [2, 6, 24, 28]. From the model perspective, budgeted training searches the optimal number of model parameters within the given compute budget [17, 26, 31, 38]. Beyond data and model perspective, optimization based methods guide budgeted training based on learning rate schedules [5, 30, 43, 44], batch size [17] and other weight averaging method [26, 27].

Among these three aspects, optimization-based methods are particularly aligned with the objectives of budgeted-iteration training. In this domain, learning rate scheduling based methods are particularly aligned with the objectives of budgeted-iteration training. Smith et al. [43] propose a new learning rate schedule, named cyclical learning rates (CLR). It improves accuracy in fewer iterations without tuning and is relevant to super-convergence phenomenon [44]. Li et al. [30] introduce an alternative setting of existing learning rate schedules for budgeted training. Chen et al. [5] propose the reflected exponential schedule (REX) via a profile and sampling fashion. Learning rate based approaches achieve robust and high-performing results under various lengths of training iterations, which corresponds to our purpose in this work, i.e. budgeted-iteration training. Besides, this approach is plug-and-play, requiring no substantial alterations to the underlying model structure, making it readily adaptable to various deep network frameworks. Therefore, we explore the proper learning rate schedule to achieve budgeted-iteration training.

Learning rate schedule The learning rate plays a pivotal role in controlling the non-convex optimization process during network training. Generally, the learning rate is gradually adjusted progressively alongside the iterations, which can be represented as $\eta = g(t)\eta_0$, where $g(t)$ is the schedule function which control the variation of learning rate, t is the current iteration and η_0 is the initial learning rate.

Learning rate schedules have been extensively studied to improve training efficiency and model performance. The common scheme is the step decay schedule. Its schedule function can be represented as $g(t) = d_{\lfloor t/t_s \rfloor}$, where t_s is the predefined decaying step and $d_i (d_i \geq d_{i+1})$ is predefined decaying scalar. A typical instance decreases the learning rate by a decaying scalar 0.1 after 50% epochs and by a decaying scalar 0.01 after 75% epochs [21]. Then Loshchilov et al. [33] observe that sharp decreases may prevent models from escaping local minima. Thus they propose cosine schedule function as $\frac{1}{2}(1 + \cos(\frac{t\pi}{t_s}))$, where t_s is the predefined stage to restart the learning rate schedule, which has proven effective in transformer training. In addition, cosine schedule is a commonly-used schedule nowadays.

Pan et al. [36] propose Eigencurve, and demonstrate that minimax-optimal convergence can be achieved for quadratic objectives when the Hessian's eigenvalue distribution exhibits substantial skewness. Pan et al. [37] further introduce a learning rate schedule called elastic step decay as $\eta_t = \eta_0/2^k$, if $t \in [(1 - r^k)T, (1 - r^{k+1})T)$ to accelerate the pre-training for Berts, where r and k are hyper-parameters that control learning rate interval. Defazio et al. [11] developed the theoretically-grounded framework for learning rate scheduling, establishing optimal linear decay schedules and refinements through rigorous mathematical derivation. Their work additionally provides the most extensive empirical evaluation of scheduling approaches to date. Hu et al. [25] propose a Warmup-Stable-Decay(WSD) for large model pretraining and promoting continuous training. WSD consists of three consecutive phases: warmup, stable constant, and decay. Luo et al. [34] discover a new schedule that outperforms the cosine schedule, resembling WSD but with lower final loss.

Adaptive learning rate Compared to schedule methods, adaptive learning rate based methods focus on the learning rate adaptation based on local loss landscape statistics. Typical methods includes

AdaGrad [13], Adadelta [54], RMSprop [23] and Adam [29]. These methods have been shown stabilizing the training process while effectively optimizing learning rate. Then Loshchilov et al. [32] propose AdamW optimizer which improves the performance in the transformer training. Recently, Chen et al. [7] discovers Lion optimizer for more memory-efficiency. Xie et al. [52] propose the Adan optimizer which implements 50% faster training than Adam and AdamW on commonly-used networks. Despite these improvements, some studies suggest that adaptive methods may underperform momentum SGD in terms of generalization [30, 51], a critical factor for budgeted-iteration training.

Learning rate design is not only significant in general training, but also critical in budgeted-iteration training which still remains a topic of debate. Some analyses advocate for small, constant learning rates to ensure stability and convergence [12]. On the contrast, one prevailing hypothesis suggests that large learning rates may facilitate crossing over sharp local minima in the optimization landscape [56]. Despite the lack of comprehensive theoretical explanations, a range of learning rate schedules inspired by the above analyses as heuristic guidelines has been widely adopted in practice, using variable learning rates to budgeted-iteration training [5, 30]. In this work, we explore learning rate schedule from optimization problem tailored to budgeted-iteration training, aiming to balance iteration budget constraints and generalization.