DYNAMIC TUNING OF WEIGHT DECAY: STABILITY ANALYSIS AND EMPIRICAL SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

Weight decay is often held constant while the learning rate varies during training, although both affect optimization dynamics. In this paper, we investigate this interaction on two levels: static hyperparameter coupling between training configurations, and the potential performance gains from dynamic weight decay scheduling. Specifically, under uniform stability analysis, we characterize how weight decay contracts perturbations in stochastic gradient descent (SGD) and changes the learning-rate conditions for stability. Based on this, matching stability bounds under a heuristic regularization-equivalence hypothesis suggests the coupling $\lambda\eta \approx C/T$, relating weight decay λ, learning rate η, and the number of updates T. For a fixed number of training epochs, this coupling further suggests that the product $\lambda\eta$ scales with batch size B. Motivated by this inverse relationship, we examine two weight-decay schedules, linear-up and iso, that increase λ as η decays while leaving the learning-rate schedule unchanged. In our experiments, hyperparameter sweeps on image-classification tasks and in Qwen3-0.6B for LoRA fine-tuning show trends consistent with the proposed coupling. Scheduling experiments across architectures and tasks with SGD show higher reported accuracy with increasing weight-decay schedules than with fixed weight decay in several settings. These findings support considering weight decay jointly with the learning rate when choosing its magnitude and scheduling it during training.

1 INTRODUCTION

The immense success of deep learning is largely attributed to its remarkable generalization capability. To further enhance this capability, techniques such as weight decay Foster (1961); Hoerl (1962); Hoerl & Kennard (1970); Krogh & Hertz (1991) and model averaging Szegedy et al. (2016); Izmailov et al. (2018) have become standard practices in modern neural network training. However, in practice, configuring these hyperparameters often relies on computationally expensive grid searches or empirical heuristics. While existing scaling laws have extensively explored the quantitative interplay between the learning rate (η) Zhou et al. (2026b) and batch size (B) McCandlish et al. (2018); Zhou et al. (2026a), λ is often tuned via expensive grid searches or, worse, treated as a fixed default value (e.g., 10^{-4}), regardless of changes in compute budget or training duration. This static treatment ignores a fundamental reality: weight decay actively alters the optimization landscape and the effective step size, necessitating a system-wide readjustment of the entire hyperparameter configuration. As shown on the left of Figure 1, in the ResNet-18/CIFAR-100 sweep shown, adding weight decay changes both the best observed test accuracy and the learning rate at which it is attained for SGD and SGD with momentum (SGDM). These observations motivate our first question:

> ***Q1***. *How does weight decay affect perturbation growth during optimization, and how does it change the sufficient learning-rate conditions for stability?*

In practice, many assume that setting λ to a fixed value and adjusting η is sufficient. However, several studies have examined this interaction through optimization dynamics D'Angelo et al. (2024), rotational equilibrium Kosson et al. (2024), and the averaging timescale of AdamW Wang & Aitchison (2025). These perspectives gradually confirm that λ should be considered together with the optimizer and its hyperparameters. Our empirical sweeps in the vision setting shown in Figure 1 suggest an approximately inverse relationship between λ and η for SGD with weight decay (SGD-WD):

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

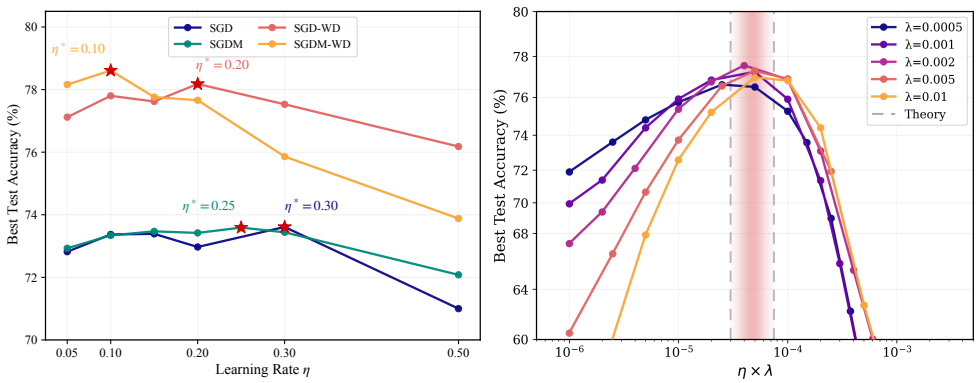


Figure 1: Coupled effect of η and λ. Left: the comparison of SGD, SGD-WD, SGDM, and SGDM-WD in optimal accuracy and corresponding learning rates. Right: at optimal test performance, the product $\lambda \times \eta$ is confined to a suitable range; each curve shows ResNet-18 performance on CIFAR-100 with a fixed λ.

Table 1: Comparison of generalization bounds for SGD and SGD-WD. Here, T represents the number of iterations, and n denotes the size of the dataset. G and L are Lipschitz constant. Our theoretical analysis shows that introducing weight decay significantly tightens the generalization bound and offers new constraints on the learning rate to achieve the optimal result.

ALGORITHM	UPDATE STRATEGY	LEARNING RATE	GENERALIZATION BOUND
SGD	$\theta_{t+1} = \theta_t - \eta\nabla_\theta F(\theta_t)$	$\eta \leq \frac{2}{L}$	$2\eta TG^2/n$ HARDT ET AL. (2016)
SGD-WD	$\theta_{t+1} = (1-\lambda\eta)\theta_t - \eta\nabla_\theta F(\theta_t)$	$\eta \leq \frac{2}{2\lambda+L}$	$2G^2/\lambda n$ THEOREM 4.6

high-performing configurations concentrate within a range of their product $\lambda\eta$ (right panel). This observation motivates studying the second question:

> ***Q2***. *What quantitative coupling between weight decay λ and other hyperparameters is suggested by stability analysis? Can this coupling guide the design of dynamic weight-decay schedules under a fixed learning-rate trajectory?*

To address **Q1**, we use uniform stability Hardt et al. (2016) to characterize how λ, η, and T affect the generalization bounds for SGD. Using the non-expansiveness of the update map, we develop a recursive analysis that accounts for gradient perturbations and λ's role in controlling weight growth. The analysis characterizes how λ affects the bounds and the associated learning rate conditions. To address **Q2**, we introduce *regularization equivalence* as a stability-matching heuristic. Matching the stability bounds of SGD-WD and SWA suggests $\lambda\eta \approx C/T$, where T is the number of parameter updates and C collects problem-dependent factors. This comparison concerns stability upper bounds and does not establish equal test performance. For a dataset of n examples trained for T_{epo} epochs with batch size B, substituting $T \approx (n/B)T_{\text{epo}}$ yields $\lambda\eta \approx CB/(nT_{\text{epo}})$. With n, T_{epo}, and C held fixed, this heuristic predicts that the product $\lambda\eta$ increases in proportion to batch size. Guided by the inverse coupling between λ and η, we examine two increasing weight-decay schedules under a shared learning rate trajectory: *linear-up* increases the coefficient linearly with training progress, whereas *iso* follows a capped inverse-learning rate rule. We compare these schedules with fixed weight decay. Schedule amplitudes are normalized to match the nominal cumulative decay strength, enabling comparisons of temporal allocation at a common nominal strength. The increasing schedules improve performance over fixed weight decay in several evaluated settings, while their relative performance depends on the optimizer, architecture, and task. These findings support using the coupling to guide schedule design and selecting the schedule shape and strength for the intended training setting.

We examine the proposed coupling trends across image-classification tasks and Qwen3-0.6B LoRA fine-tuning, and compare increasing weight-decay schedules with fixed weight decay using SGD and SGDM. With the learning-rate trajectory and nominal cumulative decay strength held fixed, increasing schedules improve test accuracy in several evaluated settings, suggesting that the temporal allocation of weight decay affects performance. Our main contributions are as follows:

- We derive stability bounds for SGD-WD based on stability as shown in Table 1. Our analysis shows that weight decay tightens the generalization bound and imposes stricter learning rate conditions for SGD-WD.
- We formulate a stability-matching heuristic that suggests $\lambda\eta \approx C/T$, derive the corresponding dependence relation $\lambda\eta \approx CB/(nT_{\text{epo}})$ under a fixed epoch budget. This rule justifies empirical observations of coupling among hyperparameters.
- We compare linear-up and iso with fixed weight decay under an unchanged learning-rate trajectory and matched nominal cumulative strength. The results show that temporal allocation can affect performance, but that its effect depends on the training setting.
- We evaluate the static coupling and dynamic schedules across image-classification models and Qwen3-0.6B LoRA fine-tuning, including settings where increasing schedules improve performance and settings where their advantage is weak or absent.

2 Related work

Weight Decay and Hyperparameter Coupling. Weight decay is coupled to learning rate, batch size, and training duration (Wang & Aitchison, 2025; Bergsma et al., 2025). We examine these relations through stability analysis and study increasing weight-decay schedules under a fixed learning-rate trajectory, with their nominal cumulative decay strength matched across schedules.

Weight Decay, Optimization, and Generalization. Weight decay affects generalization through both regularization and optimization dynamics (Zhang et al., 2019; D'Angelo et al., 2024). Building on algorithmic stability (Hardt et al., 2016), we analyze its effect on perturbation growth in SGD and derive corresponding sufficient learning-rate conditions for stability.

We provide detailed discussions and additional related works in Appendix A.

3 Preliminary

3.1 Problem Setup

Let $F(\theta, z)$ be a loss function that measures the loss of the predicted value of the parameter θ at a given sample z. There is an unknown distribution $\mathcal{D}$ over examples from some space $\mathcal{Z}$, and a sample dataset $S = (z_1, z_2, ..., z_n)$ of n examples i.i.d. drawn from $\mathcal{D}$. Then the *population risk* and *empirical risk* are defined as

$$\textbf{Population Risk: } \min_{\theta}\{R_{\mathcal{D}}[\theta] = E_{z\sim\mathcal{D}}F(\theta; z)\}; \quad \textbf{Empirical Risk: } \min_{\theta}\{R_S[\theta] = \frac{1}{n}\sum_{i=1}^{n} F(\theta; z_i)\}.$$

We define the generalization error of a model θ is as $\epsilon_{gen} = R_{\mathcal{D}}[\theta] - R_S[\theta]$. Moreover, we assume function F satisfies the following *Lipschitz* and *smoothness* assumption.

Definition 3.1 (Lipschitz)**.** For a fixed parameter $z \in Z$, a function $F(u, z) : u \in \Omega \subset R^n \to R$ is said to be *Lipschitz continuous* if there exists a constant $G \geq 0$ such that

$$|F(u; z) - F(v; z)| \leq G\|u - v\|, \quad \forall u, v \in \Omega, \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm. And it implies that $\|\nabla F(u, z)\| \leq G$ if $F \in C^1(\Omega)$.

Definition 3.2 (L-smooth)**.** For a fixed parameter $z \in Z$, a function $F(u, z) : u \in \Omega \subset R^n \to R$ is said to be *L-smooth* if there exists a constant $L \geq 0$ such that

$$\|\nabla F(u, z) - \nabla F(v, z)\| \leq L\|u - v\|, \quad \forall u, v \in \Omega, \tag{2}$$

where L is uniform for z and $\|\cdot\|$ be Euclidean norm.

Definition 3.3 (μ-strongly convex)**.** A differentiable function $F : \Omega \to R$ is said to be *μ-strongly convex* if there exists a constant $\mu > 0$ such that

$$F(u) \geq F(v) + \langle \nabla F(v), u - v\rangle + \frac{\mu}{2}\|u - v\|^2, \quad \forall u, v \in \Omega. \tag{3}$$

Definition 3.4 (convex function)**.** A differentiable function $F : \Omega \to R$ is said to be *convex* if satisfies

$$F(u) \geq F(v) + \langle \nabla F(v), u - v\rangle, \quad \forall u, v \in \Omega. \tag{4}$$

3.2 Weight Decay and Weight Averaging

SGD as a basic algorithm for solving the empirical risk minimization problem, is defined as

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta F(\theta_t, z_{i_t}), \tag{5}$$

where $\eta > 0$ is the learning rate, $\nabla_\theta F(\theta_t, z_{i_t})$ denotes the stochastic gradient w.r.t. θ and z_{i_t} is the sample chosen from dataset S by sampling with replacement in iteration t.

Weight Decay for the basic optimizer can be expressed as

$$\theta_{t+1} = (1 - \lambda\eta)\theta_t - \eta u_t, \tag{6}$$

where $\lambda \in (0, 1)$ is the weight decay coefficient. If $u_t = \nabla_\theta F(\theta_t, z_{i_t})$, then Eq.6 degenerates into SGD with L_2 regularization.

SWA obtains flatter solutions on the landscape by averaging points along the SGD full training trajectory, with the formula defined as

$$\bar{\theta}_0 = \theta_0, \qquad \bar{\theta}_T = \frac{1}{T+1}\sum_{t=0}^{T}\theta_t = \frac{T\bar{\theta}_{T-1} + \theta_T}{T+1}, \quad T \geq 1. \tag{7}$$

where θ_T is the T-th parameter in the SGD training process. This algorithm Izmailov et al. (2018) has become a standard tool used to enhance generalization performance.

4 Generalization Analysis

Q1: How does weight decay affect perturbation growth during optimization, and how does it change the sufficient learning-rate conditions for stability?

4.1 Stability and Generalization Definition

This section derives generalization bounds for various algorithms via *uniform stability*. We start by outlining the theoretical connection between generalization and stability. First, we introduce the following notions of *uniform stability* and *the expected generalization error*.

Definition 4.1 (ϵ-Uniformly Stable)**.** A randomized algorithm A is ϵ-uniformly stable if for all data sets $S, S' \in Z^n$ such that S and S' differ in at most one example, we have

$$\sup_{z \in Z} \{\mathbb{E}_A [F(A_S; z) - F(A_{S'}; z)]\} \leq \epsilon. \tag{8}$$

We define the expected generalization error of a model $\theta = A_S$, obtaining via a randomized algorithm A, as $\mathbb{E}_{S,A}[R_S[A_S] - R_{\mathcal{D}}[A_S]]$. By leveraging *uniform stability*, Hardt et al. (2016) show that stability bounds control the expected generalization error. This allows us to study the generalization properties of algorithms based on stability. We include this theorem in Appendix E.1.

Stability investigates the impact of input perturbations on model outputs. To characterize this, we analyze the update states of the original and perturbed sequences. Let S and S' denote two datasets differing by a single sample. After running the optimizer for T steps on S and S', the resulting parameters are denoted as θ_T and θ'_T. Therefore, $\mathbb{E}|F(\theta_T; z) - F(\theta'_T; z)|$ reflects the algorithm's stability under perturbations. We employ the Lipschitz property of the loss function and then obtain

$$\mathbb{E}|F(\theta_T; z) - F(\theta'_T; z)| \leq G\mathbb{E}\|\theta_T - \theta'_T\| \tag{9}$$

for all θ_T and θ'_T. This implies that we will further transform the problem of obtaining the generalization bound via the stability bound into one of controlling parameter differences.

Next, we consider the algorithm's expansion properties and establish stability bounds based on them.

Lemma 4.2. (Hardt et al. (2016, Lemma 3.6)) *Assume that the function F is L-smooth and convex. Then for any $\eta \leq \frac{2}{L}$, we have $\|\theta_{T+1} - \theta'_{T+1}\| \leq \|\theta_T - \theta'_T\|$.*

Remark 4.3. Lemma 4.2 indicates that when a convex function satisfies smoothness, and the step size is sufficiently small, perturbations exert minimal influence on the optimizer, and gradient updates exhibit non-expansive behavior. The proof is detailed in Appendix E.1. Further results can be found in papers (Hardt et al., 2016; Xiao et al., 2022).

4.2 GENERALIZATION BOUNDS FOR ALGORITHMS

Lemma 4.4. (Hardt et al. (2016, Theorem 3.8)) *Assume that the loss function $F(\theta; z)$ is convex, Lipschitz and L-smooth for all given $z \in \mathcal{Z}$ with sizes n. Suppose we run SGD with learning rate $\eta \leq \frac{2}{L}$ for T steps, where each step samples z_i from $\mathcal{Z}$ uniformly with replacement. Then, SGD has uniform stability of*

$$\epsilon_{gen} = \mathbb{E}|F(\theta_T; z) - F(\theta'_T; z)| \leq \frac{2\eta G^2 T}{n}. \tag{10}$$

Remark 4.5. Lemma 4.4 establishes a generalization upper bound dependent on the learning rate η and the number of iterations T, facilitating the analysis of optimal parameter settings. Furthermore, given that Zhang et al. (2022) identified this upper bound $\mathcal{O}(T/n)$ as also being the lower bound, it provides a theoretical foundation for comparing the generalization improvements of decay algorithms.

Theorem 4.6. *Assume that the loss function $F(w; z)$ is convex, Lipschitz and L-smooth for all given $z \in \mathcal{Z}$ with sizes n. Suppose we run SGD-WD with weight decay coefficient $\lambda \in (0, 1)$ and learning rate $\eta \leq \frac{2}{2\lambda + L}$ for T steps, where each step samples z_i from $\mathcal{Z}$ uniformly with replacement. Then, SGD-WD has uniform stability of*

$$\epsilon_{gen} = \mathbb{E}|F(\theta_T; z) - F(\theta'_T; z)| \leq \frac{2G^2}{\lambda n}. \tag{11}$$

Remark 4.7. Theorem 4.6 presents a bound $\mathcal{O}(1/n)$, which is tighter than the bound $\mathcal{O}(T/n)$ in Lemma 4.4. Unlike the linear growth with respect to iterations T in SGD, SGD-WD achieves a smaller constant upper bound, indicating that weight decay significantly improves generalization. This indicates that weight decay is not merely a regularization technique; rather, it alters the optimization dynamics, steering the algorithm toward regions with improved stability. This insight underlies our connection to SWA, which similarly seeks flatter, more stable regions of the landscape.

Remark 4.8. The tighter bound of SGD-WD is primarily attributed to the weight decay, which induces the $(1 - \lambda\eta)$-expansive property in the algorithm, denoted $\|\theta_t - \theta'_t\| \leq (1 - \lambda\eta)\|\theta_{t-1} - \theta'_{t-1}\|$. The analysis uses the sufficient step-size condition $\eta \leq 2/(L + 2\lambda)$, compared with $\eta \leq 2/L$ for SGD. The preferred learning rates and test accuracies shown in Figure 1 (left) are empirical observations that align with the theoretical analysis. For the detailed proof, please refer to Appendix F.1.

Answer to Q1: As shown by Lemma 4.4 and Theorems 4.6, weight decay not only improves the generalization of base optimizers but also imposes new requirements on learning-rate settings. Specifically, SGD-WD shows better performance than SGD, and its learning rate at optimal performance is smaller than that of SGD, which is verified in Figure 1 (left). This prompts us to rethink the coupling between λ and η.

5 GENERALIZATION-ORIENTED PARAMETER COUPLING RELATIONSHIP

> **Q2:** What quantitative coupling between weight decay λ and other hyperparameters is suggested by stability analysis? Can this coupling guide the design of dynamic weight-decay schedules under a fixed learning-rate trajectory?

Building on the "Regularization Equivalence" hypothesis proposed in this work, we seek to quantify the interplay between hyperparameters through the lens of Uniform Stability. We use stability-bound matching as a heuristic for relating weight decay, learning rate, and training duration. Specifically, we compare the upper bounds for SGD-WD and SWA. Matching these bounds provides a proxy for choosing a regularization scale; it does not imply equal stability or equal test performance.

We interpret this equivalence through the lens of regularization strength: different algorithms are considered equivalent if they induce identical generalization performance. Intuitively, explicit weight decay restricts the solution space, while implicit methods such as SWA average the trajectory to find flat minima. If both methods are tuned to yield optimal generalization, we posit that they must impose comparable constraints on the algorithm's stability. To help clarify, here are two examples:

Case 1: Different approximation mechanisms can yield the same functional outcome. For example, a target function may be approximated using either a Taylor expansion or wavelet bases; although the representations differ, their approximation capabilities are essentially equivalent.

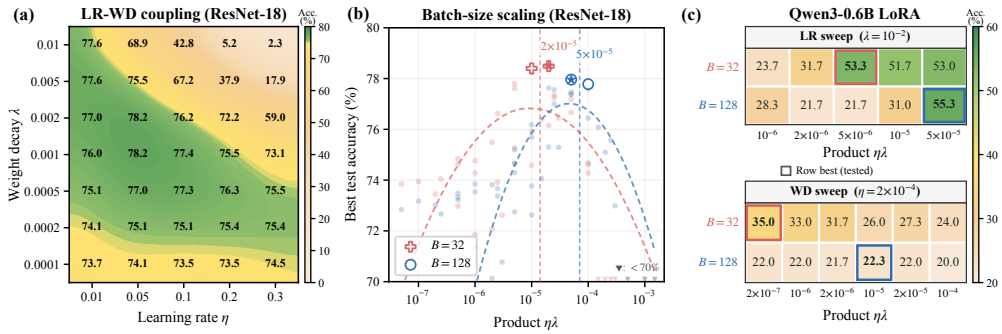


Figure 2: Hyperparameter coupling across vision and LoRA fine-tuning tasks. (a) Test accuracy over the η–λ grid for ResNet-18 on CIFAR-100. (b) Accuracy versus $\eta\lambda$ at two batch sizes; dashed curves provide visual trend guides. (c) SVAMP 0-shot accuracy for Qwen3-0.6B LoRA, varying η at fixed λ (top) and λ at fixed η (bottom). Both matrices are indexed by $\eta\lambda$; colored outlines mark the best tested configuration in each row.

Case 2: As a simple minimization example, consider $f(x) = (x^2 - 1)^2$, whose minimizers are $x = \pm 1$. After adding weight decay $\frac{\lambda}{2}x^2$ with $\lambda = 5$, the objective admits a unique minimizer at $x = 0$, which coincides exactly with the average of the original minimizers.

5.1 Coupling of λ and η

We first present the generalized bound for the SWA.

Lemma 5.1. (Wang et al. (2024, Theorem 4.1)) *Assume that the loss function $F(\theta; z)$ is convex, Lipschitz, and L-smooth for all given $z \in \mathcal{Z}$ with size n. Suppose we run SWA with step sizes $\eta \leq \frac{2}{L}$ for T steps, where each step samples z_i from $\mathcal{Z}$ with replacement. Then, SWA has uniform stability of*

$$\epsilon_{gen} = \mathbb{E}|F(\bar{\theta}_T; z) - F(\bar{\theta}'_T; z)| \leq \frac{\eta G^2 T}{n}. \tag{12}$$

Remark 5.2. Under the stated assumptions, Lemma 5.1 gives a stability upper bound of $\eta G^2 T/n$ for SWA, compared with $2\eta G^2 T/n$ for SGD. Thus, averaging halves this particular upper bound.

By bridging the bounds of SGD-WD (Theorem 4.6) and SWA (Lemma 5.1) under the Regularization Equivalence hypothesis, we equate their right-hand sides, giving $\lambda\eta = 2/T$. Motivated by this bound-level comparison, we use the following tuning relation:

$$\lambda\eta \approx \frac{C}{T}. \tag{13}$$

Here $C = 2$ for the displayed bounds; treating C as adjustable amounts to empirical calibration.

Remark 5.3. For a fixed number of updates T, Eq. (13) suggests exploring larger λ as the η decreases. The sweeps in Figure 1 (right) and Figure 2 (left) show regions of higher accuracy among the evaluated configurations, consistent with using this relation as a tuning direction. They do not determine a universal value of C or imply that configurations with the same $\lambda\eta$ have the same performance.

5.2 Coupling of λ, η and B

For n training examples and T_{epo} epochs, batch size B gives approximately $T \approx T_{\text{epo}} n/B$ parameter updates. Holding C fixed across batch sizes, substituting this approximation into Eq. (13) gives

$$\lambda\eta \approx \frac{CB}{nT_{\text{epo}}}. \tag{14}$$

Remark 5.4. Eq. (14) reveals a fundamental compensation mechanism between explicit and implicit regularization. Small-batch training introduces significant gradient noise, acting as a strong implicit regularizer. As B increases, this noise diminishes, creating a "regularization deficit." Consequently, to conserve total stability, the explicit strength $\lambda\eta$ must be scaled up proportionally to compensate. This theoretical insight perfectly matches the empirical trends in Figure 2 (middle), where increasing B from 32 to 128 necessitates shifting the optimal $\lambda\eta$ from 2e-5 to 5e-5.

6 From Static Coupling to Dynamic Weight Decay

The inverse coupling motivates increasing weight decay as the learning rate decreases. We address this part of **Q2** by comparing weight-decay schedules under a fixed learning-rate trajectory and a matched nominal cumulative decay strength.

6.1 Increasing Weight Decay During Learning-Rate Decay

Let T_{epo} denote the number of training epochs and $e \in \{0, \dots, T_{\text{epo}} - 1\}$ the epoch index. Let K_e denote the number of planned parameter updates in epoch e. Motivated by the contraction analysis underlying Theorem 4.6, we define the nominal cumulative decay strength as $I = \sum_{e=0}^{T_{\text{epo}}-1} K_e \eta_e \lambda_e$, where η_e and λ_e are the learning rate and weight decay coefficient used in epoch e. To compare the temporal allocation of weight decay, we hold both I and the learning-rate schedule fixed across schedules. For a fixed-WD baseline with coefficient $\bar{\lambda}$, the matching condition is $I = \bar{\lambda} \sum_{e=0}^{T_{\text{epo}}-1} K_e \eta_e$. Thus, each target $\bar{\lambda}$ determines the nominal cumulative strength to be matched by the dynamic schedules. Define $m_e = \eta_e / \eta_0$ as the learning-rate multiplier relative to the initial learning rate η_0. Under the decreasing learning-rate schedule, we examine two increasing weight-decay schedules,

$$\lambda_e^{\text{linear-up}} = \lambda' \frac{e}{T_{\text{epo}}}, \qquad \lambda_e^{\text{iso}} = \frac{\lambda'}{\max(m_e, \rho)}, \qquad \rho = 0.1, \tag{15}$$

where λ' is a constant amplitude determined separately for each schedule. Linear-up starts at zero and increases linearly, with λ' as its nominal endpoint. Iso keeps $\eta_e \lambda_e = \eta_0 \lambda'$ while $m_e \geq \rho$, then caps λ_e at λ'/ρ. Linear-up tests increase weight decay as the learning rate decreases, whereas iso follows a capped inverse-learning-rate rule. We compare these schedules with fixed weight decay, $\lambda_e^{\text{fixed}} = \bar{\lambda}$. For each target $\bar{\lambda}$, we compute λ' using the matching condition I. Table 2 gives the resulting amplitudes, and all schedules are evaluated over a shared grid of $\bar{\lambda}$ values within each training setting.

Table 2: Schedule-specific amplitudes at matched nominal cumulative decay strength.

Schedul	Decay strength
Fixed	$\lambda' = \bar{\lambda}$
Linear-up	$\lambda' = \bar{\lambda} \dfrac{\sum_e K_e \eta_e}{\sum_e K_e \eta_e (e/T_{\text{epo}})}$
Iso	$\lambda' = \bar{\lambda} \dfrac{\sum_e K_e \eta_e}{\sum_e K_e \eta_e / \max(m_e, \rho)}$

6.2 Results and Discussion

Table 3 compares three models across two tasks, two optimizers, and four strengths. **Iso and linear-up each exceed fixed WD in 21 of the 24 matched-strength comparisons**. Averaged equally within each task, their gains are 0.39 and 0.34 percentage points on CIFAR-100, and 0.30 and 0.19 percentage points on GSM8K, respectively. The schedule rankings depend on the training setting. With SGD, iso ranks first at all four displayed strengths for both VGG-16 and Qwen3-0.6B, with average gains over fixed WD of 0.70 and 0.48 percentage points, respectively. In contrast, linear-up ranks first at all four strengths for ResNet-50. With SGDM, iso ranks first at three of the four strengths for each vision model, whereas linear-up is best or tied for best at three strengths for Qwen3-0.6B. At the largest displayed strength, $\bar{\lambda} = 0.05$, linear-up slightly exceeds iso on both vision models with SGDM. **These comparisons show that the preferred schedule shape depends jointly on the model, optimizer, and decay strength**.

When each schedule is evaluated at its best displayed strength, both increasing schedules exceed fixed WD in all six model–optimizer settings. With SGD, iso achieves the highest reported accuracy on VGG-16 and Qwen3-0.6B, reaching 76.12% and 54.18%, compared with 75.21% and 53.63% for fixed WD. On ResNet-50, linear-up achieves the highest accuracy of 79.96%, compared with 79.67% for fixed WD. With SGDM, iso achieves the highest accuracy on VGG-16 and ResNet-50, reaching 75.36% and 79.91%, compared with 74.90% and 79.49% for fixed weight decay; linear-up achieves the highest accuracy on Qwen3-0.6B, at 50.37% versus 49.86%. However, fixed WD also performs better at some strengths. For example, at $\bar{\lambda} = 0.0001$, both increasing schedules fall below fixed WD on Qwen3-0.6B with SGDM, and iso falls below fixed WD on ResNet-50 with SGD. But this advantage at one strength does not yield a higher best result within the displayed range.

Table 3: Weight-decay schedules on CIFAR-100 and GSM8K. CIFAR-100 entries report peak test accuracy with the accompanying variability values from the source results. GSM8K entries report mean final-checkpoint test accuracy ± sample standard deviation over three seeds. Accuracy is expressed in percent. $\bar{\lambda}$ denotes the equivalent fixed weight-decay coefficient at matched nominal cumulative decay strength. Bold marks the highest reported mean within each optimizer–strength column, including ties. To save space, let "L-up" stand for "linear-up."

(a) VGG-16 / CIFAR-100

	SGDM				SGD			
$\bar{\lambda}$	0.0001	0.005	0.05	0.01	0.0001	0.005	0.05	0.01
Fixed	70.20±0.34	74.80±0.24	74.20±0.43	74.90±0.27	70.38±0.31	75.06±0.23	74.68±0.41	75.21±0.28
Iso	**70.40**±0.29	**75.17**±0.27	74.48±0.48	**75.36**±0.25	**70.76**±0.28	**75.90**±0.26	**75.36**±0.38	**76.12**±0.22
L-up	70.32±0.32	75.04±0.23	**74.54**±0.39	75.19±0.29	70.59±0.34	75.54±0.21	75.12±0.44	75.77±0.25

(b) ResNet-50 / CIFAR-100

	SGDM				SGD			
$\bar{\lambda}$	0.0001	0.005	0.05	0.01	0.0001	0.005	0.05	0.01
Fixed	74.10±0.42	79.49±0.38	77.50±0.57	78.70±0.46	74.22±0.38	79.67±0.29	77.93±0.49	78.97±0.36
Iso	**74.42**±0.37	**79.91**±0.34	77.93±0.62	**79.21**±0.39	74.10±0.43	79.88±0.32	78.16±0.56	79.05±0.31
L-up	74.38±0.45	79.86±0.41	**78.06**±0.52	79.17±0.43	**74.36**±0.36	**79.96**±0.27	**78.24**±0.47	**79.31**±0.39

(c) Qwen3-0.6B / GSM8K

	SGDM				SGD			
$\bar{\lambda}$	0.0001	0.005	0.05	0.01	0.0001	0.005	0.05	0.01
Fixed	**49.84**±0.12	49.23±0.46	49.86±0.43	49.63±0.80	53.55±0.18	53.47±0.04	53.35±1.07	53.63±0.49
Iso	49.41±0.52	50.06±0.27	**50.11**±0.33	49.46±0.04	**53.65**±0.54	**53.96**±1.01	**54.18**±0.55	**54.11**±0.36
L-up	49.36±0.27	**50.14**±0.12	**50.11**±0.33	**50.37**±0.22	53.45±0.92	53.73±0.16	53.65±0.50	53.27±0.54

Answer to Q2: Under the stability-matching heuristic, Equations 13 and 14 suggest $\lambda\eta \approx C/T$ and $\lambda\eta \approx CB/(nT_{\text{epo}})$, respectively, where T counts parameter updates. These relations connect weight decay, learning rate, training duration, and batch size. The empirical trend is consistent with the inverse relationship between η and λ (Figure 2 left). At fixed dataset size and epoch budget, the second relation suggests that the product $\lambda\eta$ scales with batch size (Figure 2 middle and right). The inverse dependence on η motivates linear-up and capped iso as increasing λ schedules during learning-rate decay. With the η trajectory and nominal cumulative decay strength fixed, their observed improvements suggest that the temporal allocation of decay affects performance. These relations guide joint tuning and schedule design, while the choice of shape and strength remains empirical.

7 Experiments

We evaluate the theoretical relationships in Sections 4–5 in four static hyperparameter experiments and one dynamic weight-decay experiment. Exp. 1 compares the best configurations found for optimizers with and without weight decay. Exp. 2 tests the inverse η–λ coupling. Exp. 3 examines the batch-size relation under linear learning-rate scaling and the coupling study in Qwen3-0.6B LoRA fine-tuning. Exp. 4 compares fixed weight decay with two increasing schedules under a shared learning-rate trajectory and matched nominal cumulative decay strength. Appendices B and C provide detailed grids, data-processing choices, seed coverage, and robustness tables.

Exp. 1: Best observed performance and learning rate. Theorem 4.6 gives a stability upper bound and a sufficient learning-rate condition for SGD with coupled weight decay; it does not by itself guarantee a higher test accuracy. We therefore treat this experiment as an empirical check of the corresponding ordering in a finite hyperparameter search. The best plain-SGD configuration uses $\eta^* = 0.30$ and reaches 73.61%, whereas the best SGD+WD configuration uses $\eta^* = 0.25$,

$\lambda = 0.005$, and reaches 78.29% (Table 4 in appendix B). The selected SGDM configurations reach 73.47% at $\eta^* = 0.15$, without WD, and 78.61% at $\eta^* = 0.10$, $\lambda = 0.0015$ with WD. Thus, the WD configurations have lower best-observed η and higher best-observed accuracies in this search.

Exp. 2: η–λ interaction. With batch size 128 and momentum 0.9 fixed, we sweep $\eta \in [0.01, 0.3]$ and $\lambda \in [10^{-5}, 10^{-2}]$. Figure 1 (left) shows an approximately anti-diagonal high-performing region on the displayed grid. For example, high-performing cells occur near $(\eta, \lambda) = (0.20, 5 \times 10^{-4})$ and $(0.05, 5 \times 10^{-3})$, consistent with keeping $\eta\lambda$ in a moderate range. The fixed-λ views provide the same qualitative coupling, but the finite grid does not establish an exact invariant or a unique optimum. The cross-architecture tables in Appendix C show that the selected λ generally moves toward smaller values as η increases, with occasional ties and architecture-dependent optima.

Exp. 3: Batch size (B), learning rate, and weight decay coupling. We use the linear rule $\eta = 0.1B/128$. The Figure 2 (right) compares $B = 32$ and $B = 128$, while the extended robustness sweep reports $B \in \{64, 128, 256, 512\}$. Equation 14 constrains the joint product $\eta\lambda$ at fixed data size and epoch budget. In Figure 2, the high-performing region shifts toward larger values of this product as batch size increases. And we fine-tune Qwen3-0.6B with LoRA on MathInstruct and evaluate the accuracy of zero-shot exact-match on SVAMP, keeping the prompt, decoding procedure, LoRA configuration, and training schedule fixed. Figure 3 (in appendix B) presents sweeps at batch size 32 and 128. In the recorded grid, the best tested small-batch point is 53.33% at $\eta\lambda = 5 \times 10^{-6}$, whereas the best tested large-batch point is 55.33% at $\eta\lambda = 5 \times 10^{-5}$. The high-performing region therefore shifts rightward on the $\eta\lambda$ axis as batch size increases, consistent with Eq. 14.

Exp. 4: Dynamic weight decay. We compare fixed weight decay with the linear-up and capped iso schedules from Section 6. All methods use 100 epochs, batch size 128, no warmup, and the same cosine learning-rate trajectory $\eta_e = 0.1\,[1+\cos(\pi e/T_{\text{epo}})]/2$. SGD uses momentum coefficient 0 and SGDM uses 0.9. Table 3 reports peak CIFAR-100 test accuracy and three-seed mean final-checkpoint GSM8K accuracy. Across the 24 displayed schedule–strength comparisons, iso and linear-up each exceed the fixed-WD mean in 21 cells. This count is descriptive and does not imply statistical significance. The average gain is larger for iso on VGG-16 with SGD ($+0.70$ percentage points) and Qwen3-0.6B with SGD ($+0.48$ points), whereas linear-up gives the larger gain on ResNet-50 with SGD ($+0.27$ points) and Qwen3-0.6B with SGDM ($+0.36$ points). On ResNet-50 with SGDM both increasing schedules improve the displayed fixed-WD mean by 0.42 points. These patterns show that the benefit and the preferred shape depend on the optimizer and task. When each method is allowed to select its best displayed strength, an increasing schedule is the best observed method in all six model–optimizer settings: iso reaches 75.36% and 76.12% on VGG-16 with SGDM and SGD, respectively; iso reaches 79.91% on ResNet-50 with SGDM, while linear-up reaches 79.96% with SGD; and linear-up reaches 50.37% on Qwen3-0.6B with SGDM, while iso reaches 54.18% with SGD. The corresponding fixed-WD maxima are 74.90%, 75.21%, 79.49%, 79.67%, 49.86%, and 53.63%.. The evidence supports increasing schedules as useful candidates at matched nominal strength, while amplitude and schedule shape remain empirical choices.

8 Conclusion

We studied how weight decay interacts with the learning rate and training duration from a uniform-stability perspective. Under convexity, our SGD analysis shows that weight decay contracts perturbations and changes the sufficient learning-rate conditions for stability. Matching the stability bounds of SGD with weight decay and SWA under the regularization-equivalence heuristic suggests $\lambda\eta \approx C/T$ and $\lambda\eta \approx CB/(nT_{epo})$ at a fixed dataset size and epoch budget. Hyperparameter sweeps on vision models and Qwen3-0.6B LoRA fine-tuning show trends consistent with this coupling. With the learning-rate trajectory and nominal cumulative decay strength held fixed, linear-up and capped iso improve accuracy over fixed weight decay in several evaluated settings. The preferred schedule and strength vary across architectures, optimizers, and tasks.

Limitation. Although our current theoretical results are consistent with extensive empirical validation (e.g., LLMs), and the proposed coupling relation captures the fundamental dynamics of optimization, the theoretical analysis is based on strict convexity assumptions. Future work aims to rigorously extend this stability framework to non-convex settings.

AI USE STATEMENT

Generative AI tools were used to standardize the formatting of tables in this paper.

REFERENCES

Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in llm pre-training, 2025. URL https://arxiv.org/abs/2505.13738.

Charlie Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Y. Prince, Björn Deiseroth, Andres Felipe Cruz-Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-μp: The unit-scaled maximal update parametrization, 2025. URL https://arxiv.org/abs/2407.17465.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

Francesco D'Angelo, Maksym Andriushchenko, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning?, 2024. URL https://arxiv.org/abs/2310.04415.

Aaron Defazio. Why gradients rapidly increase near the end of training. *arXiv preprint arXiv:2506.02285*, 2025.

Zhiyuan Fan, Yifeng Liu, Qingyue Zhao, Angela Yuan, and Quanquan Gu. Robust layerwise scaling rules by proper weight decay tuning, 2025. URL https://arxiv.org/abs/2510.15262.

Manus Foster. An application of the wiener-kolmogorov smoothing theory to matrix inversion. *Journal of the Society for Industrial and Applied Mathematics*, 9(3):387–392, 1961. doi: 10.1137/0109031. URL https://doi.org/10.1137/0109031.

Mohammad Amin Ghiasi, Ali Shafahi, and Reza Ardekani. Improving robustness with adaptive weight decay. *Advances in Neural Information Processing Systems*, 36:79067–79080, 2023.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pp. 1225–1234. PMLR, 2016.

A.E. Hoerl. Application of ridge analysis to regression problems. *Chemical Engineering Progress*, 58(3):54–59, 1962.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. ISSN 00401706. URL http://www.jstor.org/stable/1267351.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 25333–25369. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kosson24a.html.

Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Proceedings of the 5th International Conference on Neural Information Processing Systems*, NIPS'91, pp. 950–957, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1558602224.

Aitor Lewkowycz and Guy Gur-Ari. On the training dynamics of deep networks with l_2 regularization. *Advances in Neural Information Processing Systems*, 33:4790–4799, 2020.

Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. *arXiv preprint arXiv:1910.07454*, 2019.

Zhiyuan Li, Kaifeng Lyu, and Sanjeev Arora. Reconciling modern deep learning with traditional optimization analyses: The intrinsic learning rate. *Advances in Neural Information Processing Systems*, 33:14544–14555, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. URL https://api.semanticscholar.org/CorpusID:53592270.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training, 2018. URL https://arxiv.org/abs/1812.06162.

Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning: From theory to algorithms. 2014. URL https://api.semanticscholar.org/CorpusID:265498806.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

Peng Wang, Li Shen, Zerui Tao, Shuaida He, and Dacheng Tao. Generalization analysis of stochastic weight averaging with general sampling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51442–51464. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wang24bl.html.

Xi Wang and Laurence Aitchison. How to set adamw's weight decay as you scale model and dataset size, 2025. URL https://arxiv.org/abs/2405.13698.

Jiancong Xiao, Yanbo Fan, Ruoyu Sun, Jue Wang, and Zhi-Quan Luo. Stability analysis and generalization bounds of adversarial training. *Advances in Neural Information Processing Systems*, 35:15446–15459, 2022.

Zeke Xie, Zhiqiang Xu, Jingzhao Zhang, Issei Sato, and Masashi Sugiyama. On the overlooked pitfalls of weight decay and how to mitigate them: A gradient-norm perspective. *Advances in Neural Information Processing Systems*, 36:1208–1228, 2023.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ArXiv*, abs/1611.03530, 2016. URL https://api.semanticscholar.org/CorpusID:6212000.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1lz-3Rct7.

Yikai Zhang, Wenjia Zhang, Sammy Bald, Vamsi Pingali, Chao Chen, and Mayank Goswami. Stability of SGD: Tightness analysis and improved bounds. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, pp. 2364–2373. PMLR, 01–05 Aug 2022. URL https://proceedings.mlr.press/v180/zhang22b.html.

Pan Zhou, Xingyu Xie, Zhouchen Lin, and Shuicheng Yan. Towards understanding convergence and generalization of adamw. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:6486–6493, 2024. URL https://api.semanticscholar.org/CorpusID:268732016.

Yunhua Zhou, Junhao Huang, Shuhao Xing, Yechen Zhang, Runyu Peng, Qiping Guo, and Xipeng Qiu. How to set the batch size for large-scale pre-training?, 2026a. URL https://arxiv.org/abs/2601.05034.

Yunhua Zhou, Shuhao Xing, Junhao Huang, Xipeng Qiu, and Qipeng Guo. How to set the learning rate for large-scale pre-training?, 2026b. URL https://arxiv.org/abs/2601.05049.

A Additional Related Works

Parameter settings related to weight decay. The interaction between weight decay and other hyperparameters depends on the optimizer and the decay parameterization. Decoupled weight decay separates parameter shrinkage from the adaptive gradient update, distinguishing AdamW from Adam with an L_2 penalty (Loshchilov & Hutter, 2017). Training duration is also relevant: in overparameterized image-classification experiments, the time to peak test performance scales approximately as $1/\lambda$, and as $1/(\eta\lambda)$ for vanilla SGD (Lewkowycz & Gur-Ari, 2020). An exponential-moving-average interpretation of AdamW relates the decay timescale in iterations to $\tau_{\text{iter}} \approx 1/(\eta\lambda)$ and the timescale in epochs to $\tau_{\text{epoch}} \approx B/(N\eta\lambda)$, where N is the dataset size (Wang & Aitchison, 2025). The approximately stable optimal epoch timescale reported in that work implies a longer iteration timescale for larger datasets at fixed batch size.

In LLM pre-training, the optimal decay coefficient increases approximately linearly with batch size when model size, dataset size, and learning rate are fixed, over the reported range below the critical batch size (Bergsma et al., 2025). That study also finds that the preferred normalized timescale depends on the tokens-per-parameter ratio. Decay parameterization and tuning further affect hyperparameter transfer across model widths, as studied in u-μP and layerwise scaling rules (Blake et al., 2025; Fan et al., 2025). For SGD, comparable test performance is observed along approximately constant $\eta\lambda$ in multipass image-classification experiments (D'Angelo et al., 2024). These results connect decay to learning rate, batch size, model scale, and training budget through different optimization regimes.

Weight decay in the optimization system. For scale-invariant parameters in normalized networks, weight decay controls weight norms and thereby changes the effective learning rate (Van Laarhoven, 2017). This viewpoint leads to an equivalence between weight decay and a transformed learning-rate schedule in normalized networks (Li & Arora, 2019), and to the intrinsic learning rate $\eta\lambda$ governing their stochastic training dynamics (Li et al., 2020). A complementary rotational-equilibrium analysis explains how decay can balance angular updates across neurons and layers for optimizers including Adam and SGD with momentum (Kosson et al., 2024).

Several studies also vary weight decay during training. AUTOL2 decreases regularization strength as training progresses (Lewkowycz & Gur-Ari, 2020). Adaptive weight decay balances the magnitudes of classification-loss and regularization updates to improve adversarial robustness (Ghiasi et al., 2023), while gradient-norm-aware scheduled decay addresses large gradient norms in adaptive optimization (Xie et al., 2023). More recently, an analysis of late-training gradient growth motivates decreasing the decay coefficient with the learning rate for normalized layers (Defazio, 2025). These schedules target different aspects of optimization dynamics. Our schedule comparisons examine increasing decay under an unchanged learning-rate trajectory and matched nominal cumulative decay strength.

Generalization of weight decay. Classical regularization connects norm penalties to controlling estimation error, including ridge regression (Hoerl & Kennard, 1970; Shalev-Shwartz & Ben-David, 2014). In linear networks, weight decay suppresses irrelevant weight components and can reduce the influence of noise in the targets (Krogh & Hertz, 1991). In modern neural networks, explicit regularization alone does not explain generalization (Zhang et al., 2016). Mechanistic studies identify increased effective learning rates in normalized networks and, for K-FAC, approximate input–output Jacobian regularization and reduced effective damping, with the relevant mechanism depending on the architecture (Zhang et al., 2019). Further work distinguishes the role of decay in improving optimization and implicit regularization for SGD from its effects on training stability in large-scale language modeling (D'Angelo et al., 2024). For AdamW, convergence analyses are complemented by generalization bounds based on local stochastic-differential-equation approximations and PAC-Bayesian theory (Zhou et al., 2024).

Algorithmic stability provides another connection between regularization and generalization by measuring sensitivity to changes in the training sample (Bousquet & Elisseeff, 2002). This framework yields generalization bounds for stochastic gradient methods and explicitly identifies weight decay as an operation that reduces update expansiveness (Hardt et al., 2016). We build on this perspective to compare perturbation growth in SGD and SGDM with and without decay, and use stability-bound matching to motivate hyperparameter coupling and schedule design.

B Additional Experiments

This section specifies the grids, data, evaluation rules, and static-coupling results used in the main experiments. All comparisons within a sweep use the same preprocessing, model, training budget, and evaluation protocol. Reported maxima are maxima over the displayed search grids; they are not claims about a globally optimal hyperparameter.

B.1 Common Vision Protocol

The vision experiments use CIFAR-100, with 50,000 training images and 10,000 test images in 100 classes. Training uses random crops with four-pixel padding, random horizontal flips, and normalization; test images are only normalized. The main experiments use a CIFAR-adapted ResNet-18 without the initial max-pooling layer. The common training budget is 100 epochs with batch size 128, cosine learning-rate decay, no warmup, and mixed precision. The runs are implemented in PyTorch using the cosine scheduler and automatic mixed precision (AMP). Unless stated otherwise, SGD uses momentum coefficient 0 and SGDM uses 0.9. Weight decay is coupled to the update and is applied to all model parameters. We report peak test accuracy over training. For Exp. 1–3, the compared variants are plain SGD, SGDM, SGD with coupled weight decay, and SGDM with coupled weight decay. The SGDM buffer follows the PyTorch convention $m_t = \beta m_{t-1} + (1-\beta)g_t, \theta_{t+1} = \theta_t - \eta m_t$, with the coupled decay term included in the update gradient. Learning rates are swept for every variant; weight decay is swept for WD variants, and momentum is fixed or swept as stated for each experiment. The reported configurations are the best observations on the corresponding finite grids.

B.2 Optimizer and Learning-Rate Search

For each optimizer variant, we sweep the learning rate and retain the best observed test accuracy. For WD variants, the weight-decay coefficient is also selected by the search; for momentum variants, the selected momentum is reported explicitly. The SGDM rows use different selected momentum values, so their difference is not a controlled weight-decay-only comparison.

Theorem 4.6 gives a stability upper bound and a sufficient learning-rate condition for SGD with coupled weight decay. Consistent with that condition, the best observed WD configurations use lower learning rates than their corresponding no-WD rows. The accuracy differences below are empirical observations from the finite search.

Table 4: Best observed configurations in the optimizer and learning-rate search.

Optimizer	Selected η^*	Selected WD/momentum	Test accuracy (%)
SGD	0.30	$\lambda = 0$	73.61
SGD+WD	0.25	$\lambda = 0.005$	78.29
SGDM	0.15	$\lambda = 0$	73.47
SGDM+WD	0.10	$\lambda = 0.0015$	78.61

The best plain-SGD point uses $\eta^* = 0.30$ and reaches 73.61%, whereas the best SGD+WD point uses $\eta^* = 0.25$, $\lambda = 0.005$, and reaches 78.29%. The selected SGDM configurations reach 73.47% at $\eta^* = 0.15$ with $\beta = 0.1$ and no WD, and 78.61% at $\eta^* = 0.10$ with $\lambda = 0.0015$. These values support the reported finite-search observation that adding WD is associated with a lower selected learning rate and a higher observed accuracy in this setting.

B.3 Learning-Rate–Weight-Decay Grid

We fix batch size 128 and SGDM momentum 0.9, and sweep $\eta \in [0.01, 0.3]$ and $\lambda \in [10^{-5}, 10^{-2}]$. The heatmap in Figure 1 shows an approximately anti-diagonal high-performing region. Representative high-performing cells occur near $(\eta, \lambda) = (0.20, 5 \times 10^{-4})$ and $(0.05, 5 \times 10^{-3})$. Plotting fixed-λ curves against the joint product $\eta\lambda$ gives the same qualitative alignment of low training loss and high test accuracy. The finite grid supports the direction of the inverse coupling, but does not establish an exact invariant or a unique optimum.

B.4 Batch-Size Search

We use the linear learning-rate rule $\eta = 0.1\,B/128$. The main comparison uses $B \in \{32, 128\}$, and the extended robustness study uses $B \in \{64, 128, 256, 512\}$. Equation (14) constrains the joint product $\eta\lambda$ at fixed data size and epoch budget. The extended table in Appendix C shows that the selected product is non-decreasing across the listed batch sizes for each architecture, whereas the selected λ itself need not be monotone. In the main $B = 32$ versus $B = 128$ plot, the high-performing region shifts toward larger values of $\eta\lambda$ at the larger batch size. The extended table shows the same direction: for VGG-16, $\lambda = 10^{-3}$ is selected at all four reported batch sizes, while $\eta\lambda$ increases from 5×10^{-5} at $B = 64$ to 4×10^{-4} at $B = 512$. Thus the experiment supports joint-product scaling, rather than monotonic scaling of λ alone.

Under this linear learning-rate rule, it does not imply that λ alone must increase monotonically with B. In the displayed grids, the high-performing region shifts toward larger values of $\eta\lambda$ as B increases, whereas the grid-selected λ remains architecture-dependent and can remain fixed, decrease, or increase over the tested values. Thus, the experiment supports product-level joint tuning of η and λ, rather than an independent rule that scales both parameters linearly with batch size.

B.5 Qwen3-0.6B LoRA Coupling Study

We fine-tune Qwen3-0.6B on MathInstruct with LoRA in the LLaMA-Factory framework. Examples are formatted as instruction–response pairs and optimized with the native tokenizer and a fixed maximum sequence length. The learning rate and weight decay are swept while the LoRA configuration, prompting template, decoding protocol, and remaining training settings are held fixed. The learning-rate schedule is constant and uses no warmup. We use zero-shot SVAMP exact-match accuracy of the final numeric answer, with batch sizes $B = 32$ and $B = 128$.

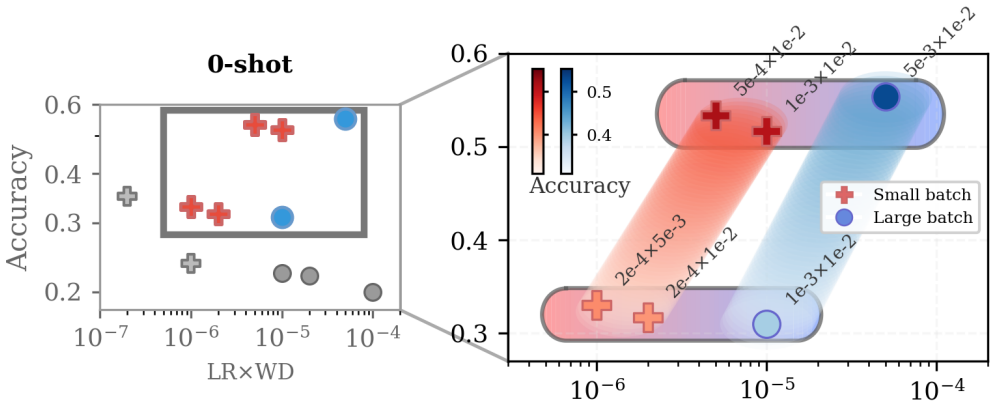


Figure 3: Qwen3-0.6B LoRA results on SVAMP under two batch sizes. The larger-batch points shift toward larger values of the joint product $\eta\lambda$ in the recorded grid.

In the recorded grid, the best small-batch point is 53.33% at $(\eta, \lambda) = (5 \times 10^{-4}, 10^{-2})$, giving $\eta\lambda = 5 \times 10^{-6}$. The best large-batch point is 55.33% at $(\eta, \lambda) = (5 \times 10^{-3}, 10^{-2})$, giving $\eta\lambda = 5 \times 10^{-5}$. These are descriptive maxima of the displayed grid; they do not imply that the product alone determines LLM accuracy.

C Robustness Across Architectures and Seeds

The robustness study repeats the static vision sweeps on ResNet-18 (11.2M parameters), VGG-16 (14.8M, without skip connections), and ResNet-50 (23.5M, deeper residual). Each architecture uses seeds $\{42, 123\}$; ResNet-18 additionally has two independent repetitions per seed. Exp. 2 uses five learning rates and seven weight-decay values. Exp. 3 uses four batch sizes and six weight-decay

Table 5: Cross-architecture comparison for the learning-rate–weight-decay grid. For each η we report the selected λ^* and peak test accuracy (%, mean $\pm$ half-range) for SGDM with $B = 128$.

	ResNet-18		**VGG-16**		**ResNet-50**	
η	λ^*	Acc. (%)	λ^*	Acc. (%)	λ^*	Acc. (%)
0.01	5×10^{-3}	77.58 ± 0.07	1×10^{-2}	74.90 ± 0.19	5×10^{-3}	79.49 ± 0.27
0.05	2×10^{-3}	78.25 ± 0.23	1×10^{-3}	73.72 ± 0.01	1×10^{-3}	77.89 ± 0.04
0.10	1×10^{-3}	77.35 ± 0.20	1×10^{-3}	73.00 ± 0.05	1×10^{-3}	77.34 ± 0.27
0.20	5×10^{-4}	76.34 ± 0.09	5×10^{-4}	72.97 ± 0.26	5×10^{-4}	77.65 ± 0.36
0.30	5×10^{-4}	75.48 ± 0.18	5×10^{-4}	72.27 ± 0.04	2×10^{-4}	77.55 ± 1.12

Table 6: Cross-architecture batch-size scaling under $\eta = 0.1B/128$. For each B we report the selected λ^* and peak test accuracy (%, mean $\pm$ half-range).

		ResNet-18		**VGG-16**		**ResNet-50**	
B	η	λ^*	Acc. (%)	λ^*	Acc. (%)	λ^*	Acc. (%)
64	0.05	1×10^{-3}	78.27 ± 0.40	1×10^{-3}	73.64 ± 0.09	5×10^{-4}	77.38 ± 0.14
128	0.10	1×10^{-3}	77.35 ± 0.20	1×10^{-3}	73.00 ± 0.05	1×10^{-3}	77.34 ± 0.27
256	0.20	5×10^{-4}	76.66 ± 0.56	1×10^{-3}	72.95 ± 0.06	5×10^{-4}	77.12 ± 0.28
512	0.40	1×10^{-3}	75.35 ± 0.44	1×10^{-3}	71.86 ± 0.20	5×10^{-4}	76.69 ± 0.00

values under the linear learning-rate rule. All other settings match the common vision protocol. The aggregate study contains approximately 640 training runs.

C.1 Cross-Architecture Learning-Rate–Weight-Decay Results

For each architecture and fixed η, Table 5 reports the selected λ^* and the corresponding peak test accuracy. The selected coefficient generally moves toward smaller values as η increases, with ties and architecture-dependent choices. Thus, these results support the direction of the inverse coupling across architectures without implying identical optima or an exactly constant product.

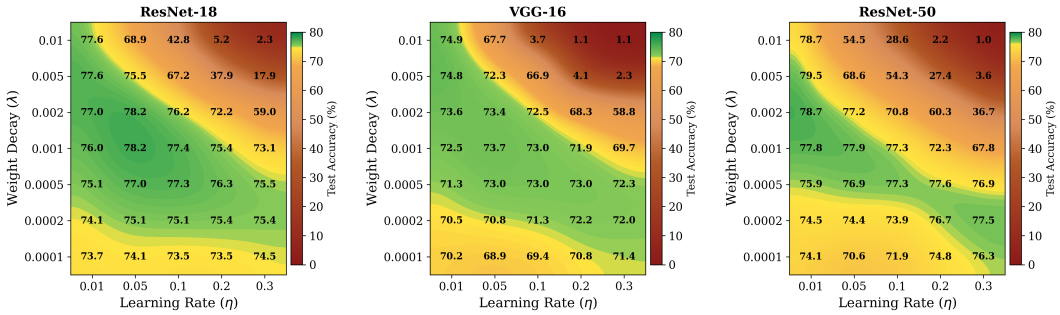


Figure 4: Full learning-rate–weight-decay heatmaps for Exp. 2 across architectures. The displayed high-performing regions are approximately anti-diagonal, with architecture-specific deviations.

C.2 Cross-Architecture Batch-Size Results

Table 6 reports the selected weight decay and peak accuracy under $\eta = 0.1B/128$. The selected $\eta\lambda$ product is non-decreasing over the listed batch sizes for each architecture, while the selected λ values vary across architectures and need not be monotone. The results therefore support the batch-dependent joint-tuning direction in Eq. (14).

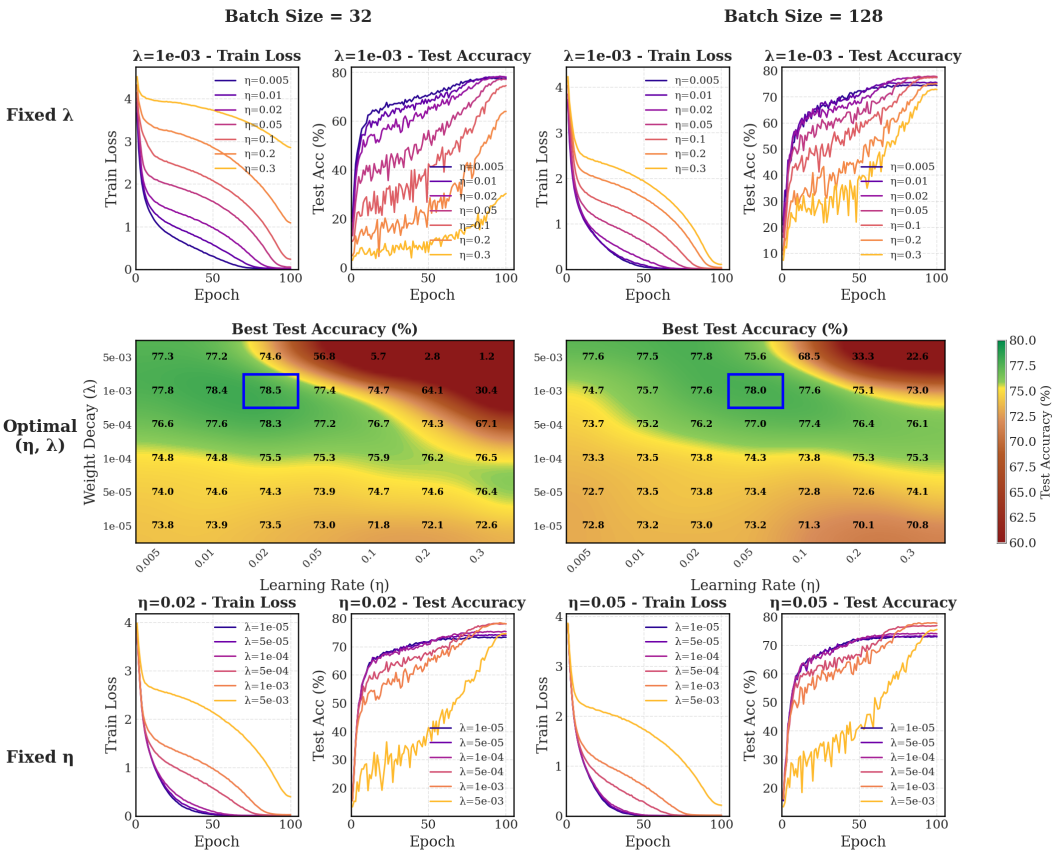


Figure 5: Batch-size coupling under the linear learning-rate rule. The panels show performance as a function of η, λ, and their joint product for $B = 32$ and $B = 128$.

D Dynamic Weight-Decay Details

The study contains three settings: ResNet-50 and VGG-16 on CIFAR-100, and LoRA fine-tuning of Qwen3-0.6B evaluated on GSM8K. We give the training protocol, baselines, schedule definitions, matching rule, and evaluation conventions for these settings.

D.1 ResNet-50 and VGG-16 on CIFAR-100

We use CIFAR-adapted ResNet-50 and VGG-16 models on CIFAR-100. ResNet-50 is the residual architecture (approximately 23.5M parameters), whereas VGG-16 has no skip connections (approximately 14.8M parameters). Training images are augmented with random crops using four-pixel padding and random horizontal flips, followed by the standard CIFAR normalization. Test images are only normalized.

Both models are trained for 100 epochs with batch size 128. All schedule conditions use the same cosine learning-rate trajectory and no warmup,

$$\eta_e = 0.1\frac{1+\cos(\pi e/T_{\text{epo}})}{2}, \qquad e = 0, \ldots, T_{\text{epo}} - 1, \tag{16}$$

where the learning rate and weight-decay coefficient are held constant within each epoch. We use coupled weight decay and evaluate both SGD, with momentum coefficient $\beta = 0$, and SGDM, with $\beta = 0.9$. This convention is used for all three schedule conditions and is distinct from applying a separate multiplicative decay after the optimizer step. Training uses mixed precision. The same optimizer implementation and learning-rate trajectory are used for all three schedule conditions.

D.2 Schedules, Baseline, and Strength Matching

The fixed-WD baseline uses a constant coefficient

$$\lambda_e^{\text{fixed}} = \bar{\lambda}.$$

The two increasing schedules use the same learning-rate trajectory:

$$\lambda_e^{\text{linear-up}} = \lambda' \frac{e}{T_{\text{epo}}}, \qquad \lambda_e^{\text{iso}} = \frac{\lambda'}{\max(m_e, \rho)}, \qquad m_e = \frac{\eta_e}{\eta_0}, \quad \rho = 0.1. \tag{17}$$

Linear-up increases the coefficient with training progress, while iso follows a capped inverse-learning-rate rule. The cap prevents the coefficient from diverging when the learning-rate multiplier becomes small. The amplitude λ' is solved separately for each schedule and each target $\bar{\lambda}$; it is not selected from the test results.

Let K_e be the number of planned parameter updates in epoch e. We match the schedules using the nominal cumulative decay strength

$$I = \sum_{e=0}^{T_{\text{epo}}-1} K_e \eta_e \lambda_e. \tag{18}$$

For a schedule shape h_e with $\lambda_e = \lambda' h_e$, the amplitude is

$$\lambda' = \bar{\lambda} \frac{\sum_e K_e \eta_e}{\sum_e K_e \eta_e h_e}. \tag{19}$$

Thus, fixed WD, iso, and linear-up share the same learning-rate trajectory and the same nominal strength within each displayed $\bar{\lambda}$ column. The quantity I is a comparison measure rather than an exact common contraction factor. In particular, for SGDM the decay term accumulates through the momentum buffer, so matching I does not make the discrete optimization trajectories identical.

For the two CIFAR-100 models, the displayed target coefficients are

$$\bar{\lambda} \in \{10^{-4}, 5 \times 10^{-3}, 5 \times 10^{-2}, 10^{-2}\}.$$

The fixed-WD run is the baseline in every column, and iso and linear-up are compared with it at the matched nominal strength. The same target grid and the same optimizer settings are used for ResNet-50 and VGG-16.

D.3 Qwen3-0.6B LoRA Fine-Tuning on GSM8K

We fine-tune Qwen3-0.6B with LoRA for the GSM8K dynamic-schedule comparison. The pretrained backbone is kept fixed and the LoRA configuration is held constant across schedules: rank 16, scaling parameter 32, and dropout 0.1. Training lasts three epochs with batch size 128, an initial learning rate of 2×10^{-4}, and cosine learning-rate decay. Input formatting, tokenization, and the remaining optimization settings are shared by all conditions.

The baseline is fixed weight decay. Iso and linear-up use the same definitions as in Eq. (17), and their amplitudes are normalized with Eq. (19) for each target $\bar{\lambda}$. The displayed target grid is $\bar{\lambda} \in \{10^{-4}, 5 \times 10^{-3}, 5 \times 10^{-2}, 10^{-2}\}$. We evaluate both SGD ($\beta = 0$) and SGDM ($\beta = 0.9$) with the coupled-decay convention above. Weight decay acts on the trainable LoRA parameters under the optimizer's parameter-group rules; the pretrained backbone remains frozen. Within each column, the comparison therefore holds the model, LoRA configuration, learning-rate trajectory, optimizer family, and nominal cumulative strength fixed while changing the temporal allocation of weight decay.

We report exact-match accuracy at the final checkpoint. The evaluation keeps the validation and test splits separate; the main table reports the test split, using 500 validation questions and 1,319 test questions. Each condition uses three random seeds, and the table reports the mean together with the sample standard deviation. The final checkpoint is used for every schedule, so no schedule or amplitude is selected from the test accuracy.

D.4 Reporting Convention

For CIFAR-100, the evaluation metric is the peak test accuracy observed over the 100 training epochs. For GSM8K, the metric is final-checkpoint exact-match accuracy, summarized across three random seeds by the mean and sample standard deviation. Boldface in the main table identifies the largest reported mean within a column; it is not a significance test. All comparisons are restricted to the displayed target-strength grid and describe schedule shape under the stated training protocol.

E Proof of Some Basic Properties

Theorem E.1. (Generalization in Expectation, Hardt et al. (2016, Theorem 2.2)) *Let A be ϵ-uniformly stable. Then,*

$$|\mathbb{E}_{S,A}\left[R_S\left[A_S\right] - R_{\mathcal{D}}\left[A_S\right]\right]| \leq \epsilon. \tag{20}$$

Remark E.2. Here, the expectation is taken only over the internal randomness of A. Theorem E.1 implies that an algorithm with uniform stability exhibits small generalization error. Therefore, the generalization bound can be determined through controlling the algorithm's uniform stability.

E.1 Proof of Lemma 4.2

***Non*-expansive.** Function is convexity and L-smoothness that implies

$$\langle \nabla F(\theta_1) - \nabla F(\theta_2), \theta_1 - \theta_2 \rangle \geq \frac{1}{L}\|\nabla F(\theta_1) - \nabla F(\theta_2)\|^2. \tag{21}$$

If $\eta \leq \frac{2}{L}$, we conclude that

$$\begin{aligned}
\|\theta_{T+1} - \theta'_{T+1}\| &= \sqrt{\|\theta_T - \eta\nabla F(\theta_T) - \theta'_T + \eta\nabla F(\theta'_T)\|^2} \\
&= \sqrt{\|\theta_T - \theta'_T\|^2 - 2\eta\langle \nabla F(\theta_T) - \nabla F(\theta'_T), \theta_T - \theta'_T\rangle + \eta^2\|\nabla F(\theta_T) - \nabla F(\theta'_T)\|^2} \\
&\leq \sqrt{\|\theta_T - \theta'_T\|^2 - \left(\frac{2\eta}{L} - \eta^2\right)\|\nabla F(\theta_T) - \nabla F(\theta'_T)\|^2} \\
&\leq \|\theta_T - \theta'_T\|.
\end{aligned} \tag{22}$$

F Generalization Bound under Convexity

F.1 Proof of Theorem 4.6

Next, we first present the generalization bound for SGD-WD based on the formulation (6):

$$\theta_t = \theta_{t-1} - \eta(\nabla F(\theta_{t-1}, z_t) + \lambda\theta_{t-1}) = (1 - \lambda\eta)\theta_{t-1} - \eta\nabla F(\theta_{t-1}, z_t). \tag{23}$$

First, we consider that the different samples are selected to update with probability $\frac{1}{n}$ at the step $T + 1$.

$$\begin{aligned}
\|\theta_t - \theta'_t\| &= \|(1 - \lambda\eta)(\theta_{t-1} - \theta'_{t-1}) + \eta(\nabla F(\theta'_{t-1}, z'_t) - \nabla F(\theta_{t-1}, z_t))\| \\
&\leq (1 - \lambda\eta)\|\theta_{t-1} - \theta'_{t-1}\| + 2\eta G,
\end{aligned} \tag{24}$$

where the inequality follows from the triangle inequality, Lipschitz assumption for the target function F.

Second, the same samples are selected with probability $1 - \frac{1}{n}$ at step $T + 1$.

$$\begin{aligned}
\|\theta_t - \theta'_t\| &= \|(1 - \lambda\eta)(\theta_{t-1} - \theta'_{t-1}) + \eta(\nabla F(\theta'_{t-1}, z'_t) - \nabla F(\theta_{t-1}, z_t))\| \\
&\leq (1 - \lambda\eta)\|\theta_{t-1} - \theta'_{t-1}\|.
\end{aligned} \tag{25}$$

This is because the objective function is convex and smooth, enabling the entire update process to exhibit $(1 - \lambda\eta)$-expansion. As shown in the following analysis: Function is convexity and L-smoothness, if $\eta \leq \frac{2}{2\lambda + L}$, we conclude that

$$
\begin{aligned}
&\|\theta_{T+1} - \theta'_{T+1}\| = \sqrt{\|(1-\lambda\eta)\theta_T - \eta\nabla F(\theta_T) - (1-\lambda\eta)\theta'_T + \eta\nabla F(\theta'_T)\|^2} \\
&= \sqrt{(1-\lambda\eta)^2\|\theta_T - \theta'_T\|^2 - 2\eta(1-\lambda\eta)\langle\nabla F(\theta_T) - \nabla F(\theta'_T), \theta_T - \theta'_T\rangle + \eta^2\|\nabla F(\theta_T) - \nabla F(\theta'_T)\|^2} \\
&\leq \sqrt{(1-\lambda\eta)^2\|\theta_T - \theta'_T\|^2 - \left(\frac{2\eta(1-\lambda\eta)}{L} - \eta^2\right)\|\nabla F(\theta_T) - \nabla F(\theta'_T)\|^2} \\
&\leq (1-\lambda\eta)\|\theta_T - \theta'_T\|,
\end{aligned}
\tag{26}
$$

where the first inequality imposes stricter requirements on the learning rate of $\frac{2\eta(1-\lambda\eta)}{L} - \eta^2 \geq 0$. Let $\delta_T = \|\theta_T - \theta'_T\|$, we obtain the expectation based on the above analysis

$$
\begin{aligned}
\mathbb{E}\left[\delta_T\right] &\leq (1 - \frac{1}{n})(1-\lambda\eta)\mathbb{E}\left[\delta_{T-1}\right] + \frac{1}{n}\left((1-\lambda\eta)\mathbb{E}\left[\delta_{T-1}\right] + 2\eta G\right) \\
&\leq (1-\lambda\eta)\mathbb{E}\left[\delta_{T-1}\right] + \frac{2\eta G}{n}
\end{aligned}
\tag{27}
$$

recursively, we can get

$$
\mathbb{E}\left[\delta_T\right] \leq \frac{2\eta G}{n}\sum_{t=0}^{T}(1-\lambda\eta)^t \leq \frac{2G}{\lambda n}. \tag{28}
$$

Plugging this back into Eq. (9), we obtain

$$
\epsilon_{gen} = \epsilon_{stab} = \mathbb{E}|F(\theta_T; z) - F(\theta'_T; z)| \leq \frac{2G^2}{\lambda n}. \tag{29}
$$

And we finish the proof.