A Better Start: Sensitivity-Aware Warm-Up for Robust and Efficient Fine-Tuning

Yile Chen[1,2], **Zeyi Wen**[2,3*], **Jian Chen**[1*], **Jin Huang**[4]

[1]South China University of Technology,
[2]The Hong Kong University of Science and Technology (Guangzhou),
[3]The Hong Kong University of Science and Technology,
[4]South China Normal University
jireh.x6@gmail.com, wenzeyi@ust.hk, ellachen@scut.edu.cn, huangjin@m.scnu.edu.cn

Abstract

As an essential component of fine-tuning, warm-up plays a crucial role in promoting stability and generalization. Many studies have examined its underlying mechanisms from different aspects. However, most of the studies focus on incorporating these insights into optimizers to reduce the reliance on warm-up. Little attention has been paid to addressing the inherent limitations of the warm-up itself, which restricts its effectiveness. In this work, we revisit warm-up from a loss landscape perspective and identify several limitations with existing warm-up, including: (1) susceptibility to nearby suboptimal traps, (2) sensitivity to hyperparameters and random seeds, and (3) inefficiency during the early stages of training. To overcome these limitations, we propose **S**ensitivity-**A**ware **W**arm-**U**p (SAWU), a lightweight and adaptive strategy that dynamically leverages learning sensitivity during warm-up to guide updates toward better and more stable basins. In addition, SAWU also introduces an adaptive scheduling mechanism and phase transition strategy across warm-up, stable, and decay phases to further enhance robustness and efficiency. Extensive experiments on various downstream tasks show that SAWU significantly outperforms the vanilla method (e.g., average 3.43% improvement on RoBerta). Moreover, SAWU can be easily combined with various optimizers and remains effective even when warm-up-based methods fail (e.g, it lifts RAdam from 49.46% to 91.78% on *qnli*). Thanks to its lightweight nature, SAWU introduces minimal overhead and even reduces training time by over 5% compared to other methods.

Code — https://github.com/JirehChan/SAWU

1 Introduction

Pre-trained language models have achieved remarkable success across a wide range of natural language processing tasks (Chang et al. 2024; Raiaan et al. 2024). Fine-tuning these models on downstream tasks has become the standard paradigm for adapting them to specific applications (Parthasarathy et al. 2024; Wu et al. 2025). Among the fine-tuning process, warm-up (He et al. 2016; Goyal et al. 2017) has emerged as an essential yet often overlooked stage. By gradually increasing the learning rate in the early iterations, warm-up improves convergence, enhances generalization, and mitigates sensitivity to initialization (He et al. 2016; Goyal et al. 2017; Gotmare et al. 2019; Zhang et al. 2023; Wortsman et al. 2024).

Despite its simplicity, warm-up has attracted increasing research attention due to its surprisingly strong empirical benefits across different tasks. A growing body of work has investigated the underlying mechanisms of warm-up, offering theoretical and empirical explanations for its effectiveness. For instance, it has been shown that warm-up helps models stay in flatter regions of the loss landscape, thereby improving generalization (Smith, Elsen, and De 2020), while also smoothing early updates and mitigating the effect of gradient noise (Gilmer et al. 2022). Building on these findings, several efforts have aimed to integrate warm-up principles into optimizer design. For instance, RAdam (Liu et al. 2020) explicitly incorporates the variance rectification mechanism of warm-up into the optimizer by introducing a term that corrects the variance of the adaptive learning rate in the early stages of training. LionA (Kosson, Messmer, and Jaggi 2024a) reduces or eliminates the need for warm-up by explicitly normalizing the update direction, thus embedding warm-up's effect of limiting large and unstable updates in the early stages of training directly into the optimizer.

However, existing work has made limited progress in improving the warm-up mechanism itself. In this work, we revisit warm-up from a landscape perspective, drawing inspiration from the concept of the safety basin (Peng et al. 2024; Chen et al. 2025)—a stable and reliable region in parameter space formed during pretraining, where the model exhibits aligned and robust behavior. During downstream fine-tuning, task-specific sub-basins are typically formed on top of this safety basin. While warm-up heuristics aim to constrain early updates within this stable region, we observe that linear schedules often fall short: they may inadvertently steer optimization toward nearby suboptimal sub-basins. This misalignment compromises both generalization and stability during the critical early stages of training. In addition, the effectiveness of warm-up is sensitive to hyperparameter choices and random seeds, often resulting in unstable convergence behavior and inconsistent performance across runs (Dodge et al. 2020; Halfon et al. 2024; Zhou, Savova, and Wang 2025). Moreover, due to its static and

*Corresponding Authors

heuristic nature, warm-up typically progresses slowly, making early training inefficient and consuming unnecessary computational resources.

To address these limitations, we propose **S**ensitivity-**A**ware **W**arm-**U**p (SAWU), a simple yet effective warm-up strategy designed to improve both stability and efficiency. SAWU incorporates learning sensitivity into the warm-up phase by monitoring how loss responds to learning rate changes, allowing the model to identify and move toward more stable and generalizable regions in the landscape. In addition, SAWU further improves fine-tuning by incorporating an adaptive scheduling mechanism and phase transition strategy to enhance efficiency and performance. In summary, our contributions are as follows:

- We revisit the role of warm-up in language model fine-tuning and analyze how it constrains optimization within safety regions of the loss landscape. We further identify key limitations of existing warm-up strategies, including their tendency to fall into nearby suboptimal basins, sensitivity to hyperparameter settings and random seeds, and inefficiency caused by rigid learning rate schedules.
- We propose Sensitivity-Aware Warm-Up (SAWU), a novel method that integrates learning sensitivity into the warm-up process. Specifically, SAWU utilizes a sensitivity score in the loss function to enhance the warm-up performance. In addition, SAWU incorporates a sensitivity-guided learning rate schedule and a phase transition mechanism, both designed to improve optimization stability and efficiency during fine-tuning.
- We conducted extensive experiments across multiple downstream tasks, demonstrating that SAWU consistently improves generalization (an average improvement of 3.43% on RoBerta) and training stability over existing fine-tuning and warm-up baselines.

2 Background and Motivation

In this section, we provide the background of our work and the motivation behind our approach. We first explain the role of warm-up in fine-tuning, then discuss its significance from the perspective of the safety basin. Finally, we point out the limitations of existing warm-up strategies, which inspired our proposed method.

2.1 Warm-Up

As a key component of fine-tuning, warm-up plays an important role in improving tuning generalization and stability (Goyal et al. 2017; Wortsman et al. 2024). By gradually increasing the learning rate at the beginning of tuning, warm-up helps stabilize early optimization and prevent the model from diverging too quickly. Meanwhile, warm-up has been shown to improve convergence, reduce sensitivity to initialization, and enhance generalization across a wide range of models and tasks (He et al. 2016; Goyal et al. 2017; Gotmare et al. 2019). Its low cost and ease of use have made it a standard component in modern fine-tuning pipelines. Researchers have explored the underlying reasons behind the effectiveness of warm-up from various perspectives, leading to numerous optimizer improvements inspired by these insights. We provide a more detailed discussion about these findings and improvements in Section 6.

The most common form of warm-up is linear scheduling, where the learning rate increases linearly from an initial value η_{init} to a target value η_{trgt} over a fixed number of steps T_{warm} as follows,

$$\eta_t = \eta_{\text{init}} + (\eta_{\text{trgt}} - \eta_{\text{init}}) \times \frac{t}{T_{\text{warm}}}, t \leq T_{\text{warm}}, \quad (1)$$

where t represents current step. In this work, we focus on linear warm-up for clarity. Beyond warm-up, fine-tuning typically follows a three-phase schedule: warm-up, stable training, and decay, each governing different learning dynamics.

2.2 Warm-up and Safety Basins

Researchers have studied warm-up from various perspectives to understand its role in model training. Prior work has explored its impact on gradient smoothness, optimization stability, and generalization behavior (Smith, Elsen, and De 2020; Gilmer et al. 2022). In this work, we revisit warm-up from a landscape perspective, drawing on the concept of the safety basin (Peng et al. 2024; Chen et al. 2025)—a region in parameter space shaped during pretraining where the model exhibits aligned, stable, and reliable behavior.

To make this perspective concrete, we visualize the loss landscape along a specific update direction from the pretrained RoBerta model, with detailed information in Appendix A. As shown in Figure 1, we plot the trajectory of SGD with and without warm-up along this direction. Without warm-up, the optimizer tends to take aggressive early steps, which may push the parameters out of the safety basin and into unstable regions. In contrast, SGD with warm-up proceeds more cautiously at the beginning, allowing the optimizer to move along a more stable path that remains within or near the low-loss region.

2.3 Limitations of Current Warm-Up

Although warm-up helps reduce the risk of fine-tuning drifting out of the safety basin and thereby enhances training stability, it still suffers from several limitations. First, vanilla warm-up lacks awareness of the surrounding loss landscape, which may cause the optimizer to settle into nearby suboptimal basins instead of exploring farther or better regions in the loss surface. Second, its behavior is highly sensitive to

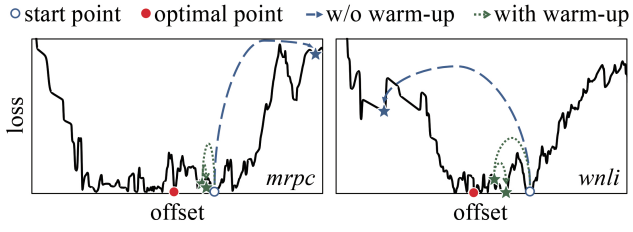

Figure 1: Fine-tuning trajectories with and without warm-up over projected loss landscape. The x-axis indicates offset along a specific update direction.

the setting of hyperparameters such as the warm-up duration T_{warm}, the target learning rate η_{trgt}, and even the choice of random seed. This sensitivity can lead to substantial variability across runs and undermine reproducibility. Third, warm-up introduces extra training overhead by delaying the use of a full learning rate in the early phase, which can slow down convergence and become increasingly costly in large-scale settings.

3 Our Method: SAWU

To overcome the limitations of the current warm-up, we propose a novel warm-up strategy called **S**ensitivity-**A**ware **W**arm-**U**p (SAWU). Instead of relying on a fixed schedule, SAWU introduces a sensitivity score that captures how responsive the loss is to changes in the learning rate. This score is used to dynamically adjust the entire fine-tuning process (i.e., warm-up, stable, and decay phases) based on training dynamics. The overall workflow of SAWU consists of three components:

- **Sensitivity-aware Loss:** monitors the model's responsiveness to learning rate changes, helping guide warm-up dynamics.
- **Sensitivity-guided Scheduling:** adaptively modulates the learning rate during both the warm-up and decay phases based on observed sensitivity.
- **Adaptive Phase Transition:** automatically determines when to exit each fine-tuning phase (i.e., warm-up, stable, and decay) by analyzing training dynamics during tuning.

In the following section, we first elaborate on each component and then provide an overview of the complete SAWU.

3.1 Sensitivity-Aware Loss

Vanilla warm-up adopts a simplistic linear increase in the learning rate, without considering the surrounding loss landscape. This lack of local sensitivity may cause the optimizer to become trapped in nearby suboptimal regions. As a result, the warm-up phase can underperform and hinder the effectiveness of subsequent fine-tuning. To address this, we propose a **sensitivity score** that measures how sensitive the loss is to changes in the learning rate, enabling more informed and effective warm-up.

Specifically, we compute the sensitivity score using a sliding window–based angular metric. After t fine-tuning steps, we obtain a sequence of learning rates $[\eta_0, \eta_1, ..., \eta_t]$ and their corresponding losses $[\mathcal{L}_0, \mathcal{L}_1, ..., \mathcal{L}_t]$, where $\mathcal{L}_i$ denotes the loss (e.g., cross entropy) after updating with learning rate η_i. For each step i ($i > 2$), we define a vector $\boldsymbol{v}_i = [\eta_i - \eta_{i-1}, L_i - L_{i-1}]$, and compute the angle θ_i between consecutive vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i-1}$. The individual score s_i is then calculated as:

$$s_i = \frac{1}{2}(1 - \cos\theta_i) = \frac{1}{2}\left(1 - \frac{\boldsymbol{v}_{i-1} \cdot \boldsymbol{v}_i}{\|\boldsymbol{v}_{i-1}\| \cdot \|\boldsymbol{v}_i\| + \epsilon}\right), \quad (2)$$

where ϵ is a small constant added for numerical stability. To better estimate the overall sensitivity, we average the scores within a sliding window of size w, resulting in the final sensitivity score:

$$S_t = \frac{1}{w-2} \sum_{i=t-w+1}^{t-2} s_i, \quad (3)$$

where $S_t \in [0, 1]$. A small S_t indicates movement along flatter and more stable regions (like direction B in Figure 2). In contrast, a high score suggests the optimizer is entering sharper or more unstable terrain (direction A), where frequent directional changes show strong sensitivity to learning rate variations.

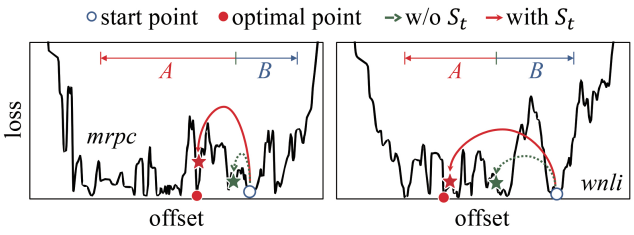

Figure 2: Warm-up trajectories with and without sensitivity score over projected loss landscape.

SAWU incorporates the sensitivity score into the loss function during the warm-up phase as,

$$\mathcal{L}_{\text{SA}} = \left(1 + \alpha \cdot (S_t - \frac{1}{2})\right) \cdot \mathcal{L}_t, \quad (4)$$

where α is a hyperparameter controlling the degree to which sensitivity influences the learning dynamics. This formulation adjusts the magnitude of the loss proportionally based on how volatile the optimization path is—amplifying the loss when the optimizer enters unstable regions (i.e., $S_t > 0.5$), and down-weighting it when the trajectory is smooth and consistent ($S_t < 0.5$). As a result, it encourages the optimizer to avoid sharp loss changes, helping it escape narrow basins near the start and explore wider, safer valleys farther away. In addition, such loss also makes warm-up more adaptive and robust, reducing sensitivity to hyperparameter choices and lowering the risk of unstable updates.

3.2 Sensitivity-Guided Scheduling

While the sensitivity-aware loss helps avoid early convergence to nearby local optima, standard fine-tuning still follows a fixed three-phase schedule (i.e., warm-up, stable, and decay) with predetermined step sizes and durations. Such rigid schedules often lead to inefficient training. To address this, SAWU introduces a sensitivity-aware scheduler that dynamically adapts learning rate in each phase based on the model's real-time sensitivity. We now detail the learning rate control strategy for each training phase.

Warm-up phase. Vanilla warm-up schedules increase the learning rate linearly, which can be either too aggressive or unnecessarily slow. Instead, SAWU gradually scales the learning rate with sensitivity-aware adjustments to better match the local landscape. Specifically, we update the learning rate as,

$$\eta_{t+1} = \eta_t + \gamma + \alpha(1 - S_t)\gamma, \quad (5)$$

where γ represents the base increment in vanilla warm-up, e.g., $\gamma = (\eta_{\text{trgt}} - \eta_{\text{init}})/T_{\text{warm}}$ in Equation (1). The weight α reuses the weighting factor from Equation (4) to maintain consistent sensitivity influence across both the loss adjustment and learning rate scaling, facilitating coordinated hyperparameter tuning. This design enables SAWU to accelerate learning rate growth in stable regions for improved efficiency, while adopting a more conservative increase in high-sensitivity areas to prevent unstable updates, thereby helping the model safely navigate the complex landscape.

Stable phase. After the warm-up phase, the learning rate is fixed, and the sensitivity score S_t becomes unavailable due to the lack of perturbations. The model trains at a constant rate to consolidate gains made during the warm-up phase. Instead of adjust learning rate during the stable phase, SAWU adaptively determines the end of warm-up to set an appropriate learning rate for stable training (cf., Section 3.3).

Decay phase. During the decay phase, the learning rate is gradually decreased with sensitivity-aware adjustments to balance convergence speed and stability. Specifically, the learning rate is updated as,

$$\eta_{t+1} = \eta_t - \gamma + \alpha(1 - S_t)\gamma \tag{6}$$

This formulation allows the decay to accelerate when the sensitivity is low (indicating a flatter and more stable loss landscape) and to slow down when the sensitivity is high, thus preventing premature reduction that could harm convergence.

Overall, this phase-aware design enables SAWU to adjust the learning rate more efficiently, avoiding unnecessary slow progression. In the following section, we describe how transitions between these phases are determined using sensitivity and loss-based criteria.

3.3 Adaptive Phase Transition

To ensure efficiency and prevent over-extending phases, SAWU applies adaptive termination rules for each stage, based on sensitivity scores or relative loss change.

Warm-up termination. In addition to the adaptive learning rate increment, SAWU introduces a sensitivity- and window-based termination criterion to avoid unnecessarily extending the warm-up phase until reaching the target learning rate η_{trgt}. Instead of relying on a fixed sensitivity threshold—which may be task-dependent and hard to tune—we monitor the trend of sensitivity over time. Specifically, the warm-up phase terminates once the absolute change in S_t remains below a small threshold β (e.g., 5%) for k consecutive steps:

$$|S_{i+1} - S_i| < \beta \cdot \min(S_{i+1}, S_i), \text{for } i = t-k, ..., t-1. \tag{7}$$

This condition shows the model has entered a stable region of the loss landscape, making further warm-up unnecessary and enabling a timely transition to the stable phase.

Stable stage termination. To determine whether to exit this stage early, SAWU introduces a simple loss-based monitor that detects stagnation in model improvement. Specifically, we track the relative change in training loss:

$$\Delta_t = \left| \frac{\mathcal{L}_t - \mathcal{L}_{t-1}}{\mathcal{L}_{t-1}} \right|. \tag{8}$$

When the loss change remains below a small threshold β (e.g., 1%) for k consecutive steps,

$$\Delta_i < \beta \quad \text{for } i = t-k, \ldots, t-1, \tag{9}$$

the model is considered to have entered a plateau region where further training at a constant learning rate is unlikely to yield significant improvements. In this case, the scheduler transitions into the decay stage.

Decay termination. To determine when to terminate the decay phase, we monitor the absolute change in training loss over consecutive steps. Once the loss difference remains below a small threshold ϵ for k steps:

$$|\mathcal{L}_{i+1} - \mathcal{L}_i| < \epsilon, \text{for } i = t-k, ..., t-1, \tag{10}$$

the model is considered to have converged sufficiently, and the decay phase ends. This early stopping criterion avoids unnecessarily prolonged training, making the schedule more adaptive and robust without requiring manual tuning.

3.4 Overview of Our SAWU

In this section, we present an overview of our proposed SAWU. The core procedure of SAWU during the warm-up phase is outlined in Algorithm 1. For completeness, the full version of the algorithm—covering the stable and decay phases—is provided in Appendix B. For better readability, we omit certain straightforward conditions such as fallback behavior when the number of training iterations is insufficient to compute the sensitivity score (i.e., smaller than the window size). In such cases, the algorithm reverts to the original loss computation (e.g., cross-entropy) and learning rate schedule (e.g., linear increase).

Algorithm 1: Basic framework of SAWU.

Input: Training data $\mathcal{D}$, model M, epoch numbers for each stage $[n_{\text{warm}}, n_{\text{stable}}, n_{\text{decay}}]$, batch size bz, initial LR η_{init}, target LR η_{trgt}, window size w, sensitivity weight α, early stop threshold β, maximum count k, threshold ϵ.

initialize: $t \leftarrow 0, \eta_1 \leftarrow \eta_{\text{init}}, n_{\text{iter}} \leftarrow \frac{|\mathcal{D}|}{bz}, \gamma \leftarrow \frac{\eta_{\text{trgt}} - \eta_{\text{init}}}{n_{\text{iter}} \cdot n_{\text{warm}}}$;

/* ***warm-up phase*** */

for i *in* $[1, n_{\text{warm}}]$ **do**

  $t \leftarrow t + 1, \mathcal{L}_t \leftarrow \text{CalLoss}(M, \mathcal{D})$;

  $s_t \leftarrow \text{CalScore}(\mathcal{L}_{t-2:t}, \eta_{t-2:t}, \epsilon)$; // Eq. 2

  $S_i \leftarrow \text{CalSAScore}(s_{t-w:t}, w)$; // Eq. 3

  $\mathcal{L}_{\text{SA}} \leftarrow (1 + \alpha(S_i - \frac{1}{2})) \cdot \mathcal{L}_i$; // Eq. 4

  $M \leftarrow \text{UpdateModel}(M, \mathcal{L}_{\text{SA}}, \eta_i)$;

  $\eta_{i+1} \leftarrow \min(\eta_i + \gamma + \alpha(1 - S_i)\gamma, \eta_{\text{trgt}})$; // Eq. 5

  if $\text{isWarmEnd}(S_{1:t}, \beta, k)$ **then**

    break; // Eq. 7

end

/* ***stable & decay phase*** */

detailed in Algorithm 2.

Output: the fine-tuned model M.

During the warm-up phase, SAWU adaptively adjusts the learning rate by analyzing the sensitivity of the model to recent optimization updates. In each iteration t, it first computes the training loss $\mathcal{L}$ (e.g., cross-entropy) on the dataset

$\mathcal{D}$ (Line 3). Then, it calculates an individual sensitivity score s_t by evaluating how recent changes in the learning rate $\eta_{t-2:t}$ affect the loss $\mathcal{L}_{t-2:t}$ over the last three steps (Line 4). Based on a sliding window of size w, SAWU further computes a smoothed sensitivity score S_i (Line 5), which is then used to scale the original loss to produce a sensitivity-aware loss $\mathcal{L}_{\text{SA}}$ (Line 6). The model M is then updated using this modified loss (Line 7). Afterward, the learning rate η_t is updated using a rule that increases it adaptively based on the current sensitivity score S_i (Line 8), and an early termination condition is checked based on the historical sensitivity trajectory $S_{1:t}$ (Lines 9-10).

After the warm-up phase, SAWU continues to employ a similar scheduling and early termination mechanism in both the stable and decay stages. During the stable stage, the learning rate is fixed at η, but the model continuously monitors the loss to determine whether an early transition to the decay stage is necessary, as described in Equation 8. In the decay stage, SAWU resumes sensitivity-aware scheduling: the learning rate is dynamically adjusted according to the computed sensitivity score, similar to the warm-up stage, but following a modified update rule (Equation 6). Early termination in the decay stage is determined based on loss changes, as formalized in Equation 10.

4 Discussion

In this section, we analyze key aspects of the proposed Sensitivity-Aware Warm-Up (SAWU). We first provide a theoretical explanation of its advantage over linear warm-up in escaping suboptimal basins. Next, we assess the computational and memory overhead to show its efficiency. Finally, we discuss the method's limitations and future directions.

- **Theoretical Insight.** We consider a stylized 1D loss landscape to simulate the fine-tuning of language model:

  $$\mathcal{L}(\theta) = ou \min_j \left\{a_j(\theta-\theta_j)^2 + b_j\right\} + c\cdot\exp(-(\theta-\theta_0)^2),$$

  where each basin (θ_j) corresponds to a local minimum, and the final term introduces a perturbation near the initialization θ_0.

  Let η_t denote the base learning rate, and $S_t \in [0,1]$ the sensitivity score measuring short-term gradient variation. SAWU defines the effective step as:

  $$\delta_t^{\text{SA}} = \eta_t \cdot (1 + \alpha(S_t - 0.5)) \cdot \|g_t\|,$$

  where $\|g_t\|$ is the Euclidean norm of gradient. In regions of high gradient volatility, S_t increases, and δ_t^{SA} is adaptively amplified. This increases the chance of crossing flat or narrow basins and prevents early convergence to nearby suboptimal θ_j.

  In contrast, linear warm-up yields $\delta_t^{\text{lin}} = \eta_t \cdot \|g_t\|$, independent of gradient dynamics. Hence, for the same η_t and $\|g_t\|$, we have:

  $$\delta_t^{\text{SA}} > \delta_t^{\text{lin}} \quad \text{if} \quad S_t > 0.5,$$

  which arises in unstable regions. Thus, SAWU adds a sensitivity-aware bias, improving escape from local minima without losing control.

- **Computational and Memory Complexity.** SAWU introduces minimal computational and memory overhead. At each step, computing the sensitivity score S_t involves only a sliding window of recent gradients and learning rates, resulting in constant-time and constant-space complexity, i.e., $\mathcal{O}(1)$ per iteration. The added operations—vector inner products and exponential scaling—are negligible compared to forward and backward passes. Thus, SAWU preserves the overall training efficiency and is scalable to large models. Moreover, thanks to its early termination mechanism across phases, SAWU can even reduce the overall training time in practice.

- **Limitations.** While SAWU demonstrates strong performance in fine-tuning language models, its design is motivated by the loss landscape characteristics (e.g., safety basin) commonly observed in such models. Its effectiveness on other tasks or architectures remains to be thoroughly evaluated. In addition, although SAWU introduces adaptivity through sensitivity-based control, it still builds upon a linear warm-up backbone and retains reliance on certain hand-tuned hyperparameters (e.g., target learning rate η_{trgt} and warm-up time $T_{\max}$). Further improvements could explore fully adaptive schedules to reduce dependence on these design choices and enhance generality.

5 Experimental Studies

To investigate the rationality of our method, we first present overall results across various downstream tasks, followed by several independent experiments (i.e., method compatibility analysis, ablation studies, and hyperparameter sensitivity analysis) to further assess key designs in SAWU. The main experimental settings are summarized below, with more detailed descriptions in Appendix C.

Datasets. We evaluated SAWU on 6 NLP tasks from HuggingFace (Wolf et al. 2020), including: *mrpc*, *qnli*, *mnli*, *qqp*, *cola*, and *stsb*.

Models. We used 2 different types of pre-trained NLP model from HuggingFace (Wolf et al. 2020), including RoBerta-base and LLaMA-3.2-1B.

Baselines. We compare against several baseline methods, including 2 popular optimizers with warm-up schedules (i.e., Adam and AdamW) and 5 warm-up-inspired optimizers, including RAdam (Liu et al. 2020), LARS (You, Gitman, and Ginsburg 2017), LAMB (You et al. 2020), RV-AdamW, and RV-Lion (Kosson, Messmer, and Jaggi 2024b).

Setup. We set the target learning rate $n_{\text{trgt}} = 1\text{e}{-4}$ and the initial learning rate $n_{\text{init}} = 1\text{e}{-6}$, with a total of 10 training epochs. For Adam, AdamW, and SAWU, we adopt a three-phase schedule with a warm-up, stable, and decay ratio of $1:4:5$. For warm-up-inspired optimizers that do not include an explicit warm-up phase, we use a $5:5$ split between stable and decay. Unless otherwise specified, SAWU is used with the Adam optimizer. Each experiment was repeated 5 times with different seeds on a Linux server with a 128-core 2.6GHz Intel Xeon Platinum 8358 CPU and 512GB RAM.

method	mrpc	qnli	mnli	qqp	cola	stsb
vanilla	83.71	50.54	35.45	63.18	0.00	87.75
+decay	84.06	50.54	31.82	67.21	34.56	89.19
+warm-up	85.86	90.24	84.44	85.89	55.75	89.87
AdamW	85.57	90.26	83.85	86.51	53.19	89.71
RAdam	86.55	49.46	81.15	63.18	61.12	90.01
LARS	66.49	86.66	81.84	85.49	48.02	83.26
LAMB	86.03	90.95	86.49	90.23	57.79	88.76
RV-AdamW	66.49	50.59	35.45	63.18	57.21	88.48
RV-Lion	66.49	49.46	35.45	63.18	56.12	88.12
ours	**87.59**	**92.62**	**87.00**	**91.01**	**63.12**	**90.26**

Table 1: Test performance (%) of full fine-tuning RoBerta.

	mrpc	qnli	mnli	qqp	avg.
Adam	87.72	91.40	88.54	91.21	89.72
AdamW	82.72	91.38	88.53	92.13	88.69
RAdam	81.45	90.55	88.71	90.39	87.78
LAMB	58.96	49.46	80.21	89.71	69.59
ours	**88.91**	**92.14**	**90.21**	**93.13**	**91.10**

Table 2: Accuracy (%) of fine-tuning LLaMA with LoRA.

5.1 Overall Experiments

To verify the effectiveness of SAWU, we first conducted experiments on multiple downstream tasks to compare its performance with various baselines in this section. Additional implementation details and experimental results are provided in Appendix C.

- **Accuracy.** As shown in Table 1 and Table 2, warm-up plays a significant role for vanilla optimizer (i.e., Adam) in fine-tuning pre-trained models. Without it, optimization can deviate from the safety basin near initialization, leading to poor performance, particularly on sensitive tasks such as *cola* and *mnli*. Although warm-up-inspired optimizers like RAdam and LAMB can achieve strong results on certain tasks, their effectiveness is inconsistent across all tasks and may require a further hyperparameter tuning. For example, RAdam performs well on *stsb* but fails on *qnli* in Table 1. In contrast, SAWU consistently outperforms all baselines. It achieves the best results on all RoBerta tasks with an average gain of +1.89%, and improves LLaMA accuracy by +1.38%.
- **Time cost.** Figure 3 presents the fine-tuning time cost for each method. While warm-up-inspired optimizers such as RAdam and LAMB can partially reduce the need for explicit warm-up, they often introduce significant time overhead. For instance, RAdam increases total training time on *qnli* by +19.2%, and LAMB by +23.6%, compared to standard Adam. In contrast, the additional overhead introduced by SAWU is negligible. Moreover, benefiting from the adaptive phase transition mechanism, SAWU even reduces training time on some tasks, achieving over 5% time savings on *mrpc* and *qnli*.

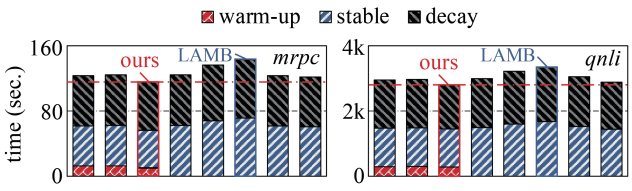

Figure 3: Fine-tuning time (sec.) for 8 methods (i.e., Adam, AdamW, ours, RAdam, LARS, LAMB, RV-AdamW, and RV-Lion).

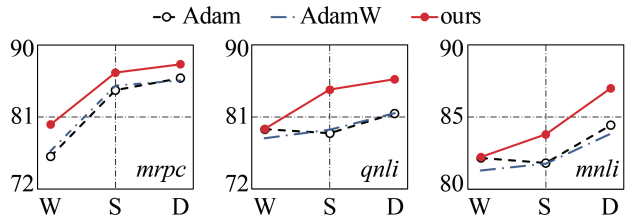

Figure 4: Test accuracy (%) across different phases (W: Warm-up, S: Stable, D: Decay).

- **Phase-wise performance.** To further understand the dynamics behind the performance gains, we visualize the accuracy trajectory of Adam, AdamW, and our method across different training phases (warm-up, stable, and fine-tuning). As shown in Figure 4, our method consistently identifies better basins during the warm-up phase. For instance, on *mrpc*, SAWU achieves over 4% higher accuracy than the vanilla optimizer at the end of the warm-up, indicating early convergence to a high-quality region. On tasks like *qnli* and *mnli*, although the warm-up accuracy is comparable across methods, the basin reached by SAWU leads to more favorable subsequent optimization, resulting in better final performance. These results confirm that sensitivity-aware warm-up enables safer and more effective initialization for downstream fine-tuning.

5.2 Independent Experiments

In addition to the overall experiment, we conducted additional independent experiments to further investigate the effectiveness and robustness of SAWU, including its compatibility with other optimizers, ablation studies of key components, and an analysis of hyperparameter sensitivity. All experiments are conducted based on full fine-tuning of RoBerta, with additional details provided in Appendix C.

- **Method compatibility.** We evaluate the compatibility of SAWU with various optimizers, as shown in Table 3. Overall, SAWU consistently improves performance across different optimizers, notably enhancing results for warm-up-inspired methods like RAdam and LAMB by mitigating their instability. While a slight drop is observed with LAMB on *mnli*, the overall results demonstrate that SAWU is broadly applicable and optimizer-agnostic.
- **Ablation studies.** To isolate the effects of the major components in SAWU, we perform ablation experiments by

	AdamW	+ours	RAdam	+ours	LAMB	+ours
mrpc	85.57	87.14$^{\uparrow}$	86.55	88.00$^{\uparrow}$	86.03	86.80$^{\uparrow}$
qnli	90.26	91.91$^{\uparrow}$	49.46	91.78$^{\uparrow}$	90.95	92.26$^{\uparrow}$
mnli	83.85	86.43$^{\uparrow}$	81.15	83.19$^{\uparrow}$	86.49	86.37$^{\downarrow}$
qqp	86.51	90.18$^{\uparrow}$	63.18	87.34$^{\uparrow}$	90.23	90.25$^{\uparrow}$

Table 3: Test accuracy (%) of fine-tuning w/ and w/o SAWU.

method	warm-up		stable		decay	
	acc.	time	acc.	time	acc.	time
vanilla	76.06	13	84.35	49	85.86	62
+SL	83.01$^{\uparrow}$	13	85.14$^{\uparrow}$	49	87.51$^{\uparrow}$	62
+SS	82.96$^{\updownarrow}$	12$^{\downarrow}$	85.32$^{\uparrow}$	49	87.62$^{\uparrow}$	61$^{\downarrow}$
+AP	82.98$^{\updownarrow}$	10$^{\downarrow}$	85.28$^{\updownarrow}$	48$^{\downarrow}$	87.59$^{\updownarrow}$	60$^{\downarrow}$

Table 4: Test accuracy (%) of fine-tuning RoBerta.

selectively removing or modifying the sensitivity-aware loss ("+SL"), scheduling mechanism ("+SS"), and adaptive phase transition ("+AP"). As shown in Table 4, the sensitivity-aware loss contributes the most to performance improvement, which aligns with our design intuition and observations. Although the scheduling adjustment slightly reduces accuracy during the warm-up phase by cutting unnecessary slow increases, it does not harm subsequent fine-tuning and can even bring slight gains, along with reduced training time. In addition, the adaptive phase transition cuts time cost without sacrificing performance.

- **Hyperparameter sensitivity.** To further investigate SAWU's sensitivity to hyperparameters, we conducted experiments varying key parameters, with results shown in Fig. 5. (1) **Window size** (w): Smaller window sizes (3–7) effectively improve warm-up performance by capturing recent learning dynamics more accurately. However, a larger window (e.g., 9) considers too long a history, which can impair decision-making and lead to accuracy drops, especially during warm-up. (2) **Sensitivity weight** (α): Moderate values around 0.5 achieve the best balance, maximizing warm-up accuracy (83.12%) and stable phase performance. Too low or too high α slightly degrades results, indicating the importance of tuning this parameter to properly weigh sensitivity in the loss. (3) **Maximum count** (k): Too small k (e.g., 3) can cause premature stopping and degrade warm-up accuracy (67.99%), while larger k values (7 or 9) yield more stable and higher accuracy. (4) **Stop threshold** (β): Intermediate values (0.05–0.1) achieve optimal results across all phases, with stable and decay accuracies peaking near 86%–87%. Values too low or too high result in suboptimal transitions and decreased performance.

6 Related Works

In this section, we provide a brief overview of prior work related to SAWU, focusing on warm-up strategies and their integration into optimization design. More comprehensive discussions can be found in Appendix D.

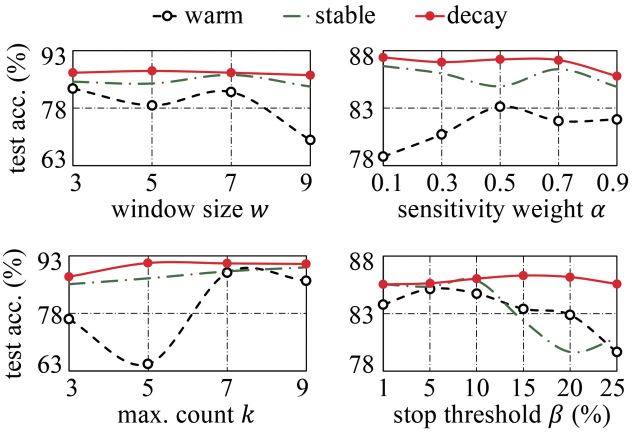

Figure 5: Hyperparameter sensitivity experimental results.

Warm-up is widely adopted as a simple yet effective technique in large-scale training, especially in NLP and vision. Its core idea is to gradually increase the learning rate during early steps, allowing smoother updates and reducing instability (He et al. 2016; Goyal et al. 2017). Beyond its success, many studies have explored warm-up's mechanisms. Some link its benefits to better generalization via flatter loss landscapes (Smith, Elsen, and De 2020), while others highlight its role in reducing gradient noise and stabilizing early optimization (Gilmer et al. 2022; Wortsman et al. 2024).

Building on these insights, recent research has incorporated warm-up principles directly into optimization strategies. Liu et al. (Liu et al. 2020) decouple warm-up from weight decay to improve training dynamics. Kalra et al. (Kalra and Barkeshli 2024) design adaptive optimizers considering warm-up effects, while Kosson et al. (Kosson, Messmer, and Jaggi 2024a) analyze learning rate ramp-up to better guide large-scale training. These efforts reflect growing interest in moving beyond static warm-up heuristics toward more adaptive, optimization-aware designs.

7 Conclusion

In this paper, we introduced Sensitivity-Aware Warm-Up (SAWU), a lightweight and adaptive fine-tuning framework that dynamically adjusts warm-up behavior based on learning sensitivity. SAWU introduces a sensitivity score into fine-tuning, which monitors how changes in the learning rate affect the loss and guides adjustments across the warm-up, stable, and decay phases. By jointly improving the loss design, learning rate scheduling, and phase transition, SAWU effectively improves training stability, generalization, and efficiency. Extensive experiments on various downstream tasks show that SAWU outperforms vanilla warm-up (e.g., +3.43% on RoBerta), integrates well with various optimizers, and remains effective when others fail (e.g., boosting RAdam from 49.46% to 91.78% on *qnli*). Despite its effectiveness, SAWU adds minimal overhead and even reduces training time by over 5%.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62376099), the Natural Science Foundation of Guangdong Province (Grant No. 2024A1515010989), and the Guangzhou Industrial Information and Intelligent Key Laboratory Project (No. 2024A03J0628).

References

Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. 2024. A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 15(3): 1–45.

Chen, H.; Dong, Y.; Wei, Z.; Huang, Y.; Zhang, Y.; Su, H.; and Zhu, J. 2025. Understanding Pre-training and Fine-tuning from Loss Landscape Perspectives. *arXiv preprint arXiv:2505.17646*.

Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; and Smith, N. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv preprint arXiv:2002.06305*.

Gilmer, J.; Ghorbani, B.; Garg, A.; Kudugunta, S.; Neyshabur, B.; Cardoze, D.; Dahl, G. E.; Nado, Z.; and Firat, O. 2022. A Loss Curvature Perspective on Training Instabilities of Deep Learning Models. In *International Conference on Learning Representations (ICLR)*.

Gotmare, A.; Keskar, N. S.; Xiong, C.; and Socher, R. 2019. A Closer Look at Deep Learning Heuristics: Learning rate restarts, Warmup and Distillation. In *International Conference on Learning Representations (ICLR)*.

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, Large Minibatch SGD: Training Imagenet in 1 Hour. *arXiv preprint arXiv:1706.02677*.

Halfon, A.; Gretz, S.; Arviv, O.; Spector, A.; Toledo-Ronen, O.; Katz, Y.; Ein-Dor, L.; Shmueli-Scheuer, M.; and Slonim, N. 2024. Stay Tuned: An Empirical Study of the Impact of Hyperparameters on LLM Tuning in Real-World Applications. *arXiv preprint arXiv:2407.18990*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Kalra, D. S.; and Barkeshli, M. 2024. Why Warmup the Learning Rate? Underlying Mechanisms and Improvements. *Advances in Neural Information Processing Systems (NeurIPS)*, 37: 111760–111801.

Kosson, A.; Messmer, B.; and Jaggi, M. 2024a. Analyzing & reducing the need for learning rate warmup in GPT training. *Advances in Neural Information Processing Systems (NeurIPS)*, 37: 2914–2942.

Kosson, A.; Messmer, B.; and Jaggi, M. 2024b. Rotational Equilibrium: How Weight Decay Balances Learning across Neural Networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, ICML'24. JMLR.org.

Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Han, J. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In *International Conference on Learning Representations (ICLR)*.

Parthasarathy, V. B.; Zafar, A.; Khan, A.; and Shahid, A. 2024. The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities. *arXiv preprint arXiv:2408.13296*.

Peng, S. Y.; Chen, P.-Y.; Hull, M.; and Chau, D. H. 2024. Navigating the Safety Landscape: Measuring Risks in Fine-tuning Large Language Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 37: 95692–95715.

Raiaan, M. A. K.; Mukta, M. S. H.; Fatema, K.; Fahad, N. M.; Sakib, S.; Mim, M. M. J.; Ahmad, J.; Ali, M. E.; and Azam, S. 2024. A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. *IEEE Access*, 12: 26839–26874.

Smith, S.; Elsen, E.; and De, S. 2020. On the Generalization Benefit of Noise in Stochastic Gradient Descent. In *International Conference on Machine Learning (ICML)*, 9058–9067. PMLR.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771.

Wortsman, M.; Liu, P. J.; Xiao, L.; Everett, K. E.; Alemi, A. A.; Adlam, B.; Co-Reyes, J. D.; Gur, I.; Kumar, A.; Novak, R.; Pennington, J.; Sohl-Dickstein, J.; Xu, K.; Lee, J.; Gilmer, J.; and Kornblith, S. 2024. Small-scale Proxies for Large-scale Transformer Training Instabilities. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Wu, X.-K.; Chen, M.; Li, W.; Wang, R.; Lu, L.; Liu, J.; Hwang, K.; Hao, Y.; Pan, Y.; Meng, Q.; et al. 2025. LLM Fine-Tuning: Concepts, Opportunities, and Challenges. *Big Data and Cognitive Computing*, 9(4): 87.

You, Y.; Gitman, I.; and Ginsburg, B. 2017. Large Batch Training of Convolutional Networks. arXiv:1708.03888.

You, Y.; Li, J.; Reddi, S.; Hseu, J.; Kumar, S.; Bhojanapalli, S.; Song, X.; Demmel, J.; Keutzer, K.; and Hsieh, C.-J. 2020. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In *International Conference on Learning Representations (ICLR)*.

Zhang, A.; Lipton, Z. C.; Li, M.; and Smola, A. J. 2023. *Dive into Deep Learning*. Cambridge University Press.

Zhou, H.; Savova, G.; and Wang, L. 2025. Assessing the Macro and Micro Effects of Random Seeds on Fine-Tuning Large Language Models. *arXiv preprint arXiv:2503.07329*.