SOLAR: A STATE-DRIVEN ONLINE LEARNING RATE SCHEDULER FOR LLM PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning-rate (LR) scheduling plays a central role in large language model (LLM) pretraining, yet current practice still relies heavily on hand-crafted heuristics such as Warmup-Cosine-Decay and Warmup-Stable-Decay. Because these schedules are fixed in advance, they cannot adapt to evolving optimization dynamics. Online learned scheduling within the Learning to Optimize (L2O) framework offers a dynamic alternative, but remains brittle at LLM scale due to noisy signals, delayed feedback, and the risk of catastrophic divergence. We propose **S**tate-driven **O**nline **L**earning r**A**te schedule**R** (SOLAR), a stabilized framework for reliable online LR adaptation. SOLAR uses a base schedule as a reference and learns bounded, state-dependent residual corrections for individual parameter groups. Each correction re-anchors to the base at every step, allowing the policy to adapt the LR without relearning the warmup–decay profile. A lightweight state representation and progress-aware reward guide online learning, while a Circuit-Breaker restores training after rare unsafe actions. Across autoregressive language-model pretraining, SOLAR improves final perplexity over tuned static schedules and automatic LR tuners for dense models from 60M to 1B, AdamW and Muon, and two MoE settings up to 3B. Matched 130M controls show that adding base anchoring and action bounds improves a global PPO controller from 27.09 to 23.74 final PPL, while group-wise control reaches 22.87 on the same two seeds. A residual policy trained on a 60M proxy can also be frozen and reused at larger dense scales without target PPO updates, remaining effective across a fourfold base-LR range. These results establish SOLAR as a practical learned LR controller for LLM pretraining.

1 INTRODUCTION

Despite decades of progress in optimization, the learning-rate schedule for pretraining billion-parameter large language models (LLMs) is still set by hand and fixed before training begins. In practice, hand-crafted schedules such as Warmup-Cosine-Decay (Cosine) (Loshchilov & Hutter, 2017) and Warmup-Stable-Decay (WSD) (Hu et al., 2024) remain the default due to their simplicity, reliability, and ease of deployment (Touvron et al., 2023a;b). Yet they impose a fundamental limitation: once training begins, the schedule is entirely predetermined and cannot adapt to the evolving optimization landscape, including shifts in loss dynamics, gradient norms, and parameter magnitudes. Adaptive optimizers do not resolve this limitation: although methods such as Adam (Kingma & Ba, 2015) and Muon (Jordan et al., 2024) rescale updates at the individual parameter level, the global learning rate that governs the overall update magnitude is still specified in advance.

A natural alternative is *online learning-rate scheduling*, where the learning rate is adjusted on the fly according to current and historical training signals (Baydin et al., 2018; Jin et al., 2021; Daniel et al., 2016). This approach aligns closely with the broader Learning-to-Optimize (L2O) paradigm, as it requires the scheduler to dynamically adapt based on the unfolding optimization state. In large-scale LLM pretraining, however, online scheduling is notoriously difficult to operationalize. The available optimization signals, such as loss trends and gradient norms, are noisy—that is, they are high-variance, non-stationary, and often only weakly informative at the level of individual steps when optimizing a nonconvex objective with stochastic methods (Chowdhery et al., 2023; Huang et al., 2025; McCandlish et al., 2018). A suboptimal scheduling decision may not trigger an immediate failure yet can silently degrade progress over a long horizon; conversely, an overly aggressive adjustment can cause abrupt instability within a few steps (Takase et al., 2025). These delayed and

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.

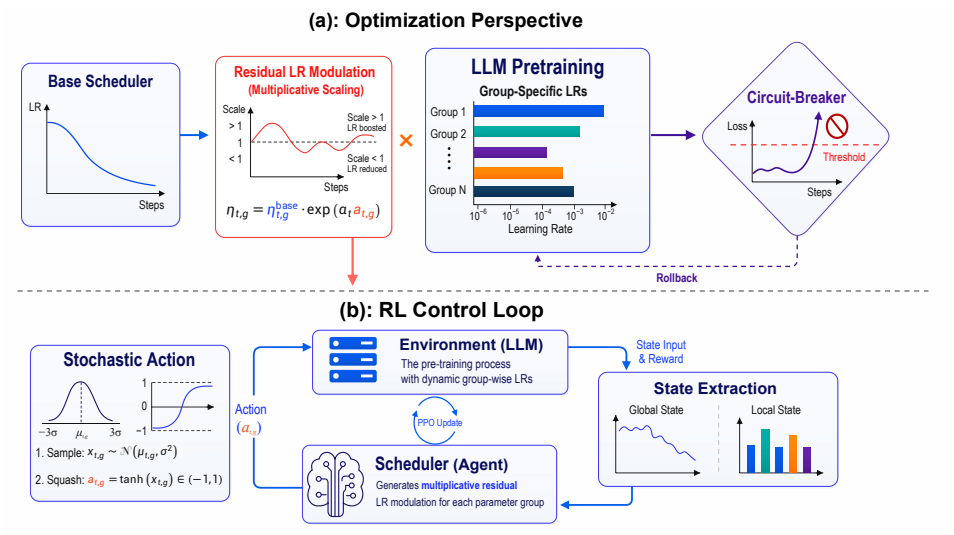


Figure 1: Overview of SOLAR. Panel (a) presents the optimization perspective: SOLAR augments a base scheduler with residual learning-rate modulation, applies bounded group-wise LR actions to the LLM pretraining process, and uses an automated Circuit-Breaker to roll back unsafe trajectories when severe loss spikes are detected. Panel (b) illustrates the RL control loop: SOLAR constructs lightweight global and local state features from the current training dynamics, outputs parameter-group-wise residual actions, receives delayed reward feedback after the optimizer update, and improves the scheduler through PPO updates.

asymmetric failure modes make learned exploration of the learning rate particularly brittle in realistic pretraining runs. As a result, although learned schedulers have shown promise in smaller settings, they have not yet been widely demonstrated as reliable solutions in realistic LLM pretraining settings. This gap motivates the search for an online scheduler that is not only adaptive but also robust enough to operate in such settings.

In this work, we introduce **SOLAR** (**S**tate-driven **O**nline **L**earning r**A**te schedule**R**), a reinforcement learning (RL)-based L2O framework for online learning-rate scheduling in LLM pretraining. Rather than generating the full LR trajectory, SOLAR uses a base schedule as a reference and learns bounded, state-dependent residual corrections for individual parameter groups. The base supplies the warmup and decay profile, while the policy adapts to the realized optimization state. Each correction re-anchors to the current base LR, preventing exploratory errors from compounding into a new schedule. As illustrated in Figure 1, SOLAR maps lightweight training-state features to parameter-group-wise LR adjustments online. Its stochastic residual actions support fine-grained exploration, and action-scale warmup limits early perturbations. A progress-aware reward combines immediate improvement, longer-term trend, and group stability. A threshold-based Circuit-Breaker detects severe loss spikes and restores the latest safe checkpoint.

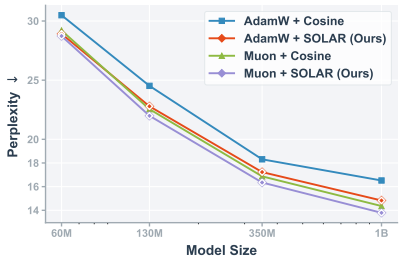


Figure 2: Pretraining performance on C4 across model scales. SOLAR consistently achieves the lowest perplexity, outperforming strong baselines in both AdamW and Muon families.

We evaluate SOLAR on autoregressive language modeling over C4 (Raffel et al., 2020) for Llama 2 (Touvron et al., 2023b) models ranging from 60M to 1B parameters, using AdamW (Loshchilov & Hutter, 2019) and Muon. For the 1B model, SOLAR reduces perplexity (PPL) by over 10% relative to AdamW with Cosine and by about 4% relative to Muon with Cosine (Figure 2). Repeated runs preserve the gains at every repeated scale, including two paired 1B AdamW runs. We further validate SOLAR on pretraining a 1B-parameter Qwen2-MoE (Yang et al., 2024) model on The Pile (Gao et al., 2020) under AdamW. SOLAR improves final perplexity from 9.61 to 9.34 while reducing early-stage optimization volatility (Figure 3); a 3B MoE run improves from 10.73 to 10.38 (Appendix C.12). Matched 130M controls clarify why the method works: a recursive global PPO controller ends at 27.09 PPL, base anchoring and action bounds improve it to 23.74,

and group-wise control reaches 22.87 on the same two seeds. A residual policy trained exclusively on a 60M proxy can also be frozen and reused at larger dense scales without target PPO updates or SOLAR-specific retuning; it remains effective across $0.5\times$–$2\times$ base LRs.

Overall, our main contributions are:

- **Methodological Design.** We propose SOLAR, an RL-based framework for online LR scheduling in LLM pretraining. SOLAR keeps the base LR profile outside the learned controller and learns bounded, state-dependent group-wise corrections that re-anchor at every step. A progress-aware reward and Circuit-Breaker support policy learning throughout a full pretraining run.
- **Empirical Demonstration.** SOLAR improves final PPL over tuned static schedules and automatic LR tuners across dense and MoE models under AdamW and Muon, with repeated gains through 1B. On matched seeds, controls show that base anchoring and action bounds improve a global PPO controller by 3.35 PPL, while group-wise control contributes another 0.87 PPL.
- **Practical Scalability.** A residual policy acquired through full-length online runs on a 60M proxy can be frozen and reused at larger dense scales without target PPO updates or SOLAR-specific retuning, while remaining effective across a fourfold base-LR range.

2 Related Work

Learning Rate Scheduling and Hyperparameter Optimization. Preset schedules encode the open-loop training profile. Cosine (Loshchilov & Hutter, 2017; Touvron et al., 2023a), WSD (Hu et al., 2024), CLR (Smith, 2017), and one-cycle schedules (Smith & Topin, 2019) choose warmup and decay before observing the realized run. Blockwise LR (Wang et al., 2025) assigns different rates to transformer module types but remains preset. Hyperparameter-optimization methods search this profile: Bayesian optimization (Snoek et al., 2012) and population-based training (Jaderberg et al., 2017) use multiple trials, while AutoLRS (Jin et al., 2021) evaluates short candidate segments online. MECHANIC (Cutkosky et al., 2023) instead adapts a global multiplier. SOLAR keeps the base profile outside the learned controller and learns closed-loop corrections within the live target run.

Learned Learning-Rate Controllers. Learned LR control has a substantial lineage. Early work uses reinforcement learning to select optimization hyperparameters or step sizes (Hansen, 2016; Daniel et al., 2016; Xu et al., 2017). Xu et al. (2019) train a PPO controller from past training histories and transfer it across image tasks. GNS (Xiong et al., 2022) encodes layer states with a graph network and learns a global schedule across target episodes. Subramanian et al. (2023) study PPO scheduling on MNIST and CIFAR, while GANNO (Tessera et al., 2023) uses layer-wise agents and absolute LR actions. These studies establish RL scheduling, state conditioning, transfer, and layer-wise control. SOLAR addresses a different operating constraint: a single full autoregressive LLM pretraining trajectory must both train the controller and remain competitive. The base schedule supplies the open-loop profile; the policy learns only bounded residual corrections that re-anchor every step. To our knowledge, SOLAR is the first learned LR scheduler shown to train online throughout the same full LLM pretraining run that it improves. Appendix C.1 compares the learned objects and acquisition protocols, and Appendix C.4 evaluates matched controllers in the LLM setting.

Adaptive and Schedule-Free Optimizers. Adaptive optimizers modify update magnitudes from gradient history. AdaGrad (Duchi et al., 2011), RMSProp (Tieleman & Hinton, 2012), AdamW (Loshchilov & Hutter, 2019), Adafactor (Shazeer & Stern, 2018), Lion (Chen et al., 2023), and Muon (Jordan et al., 2024) differ in their internal update geometry, but still accept an external LR trajectory. D-Adaptation (Defazio & Mishchenko, 2023), Prodigy (Mishchenko & Defazio, 2024), and Schedule-Free optimization (Defazio et al., 2024) reduce that dependence through optimizer-side scalar adaptation or iterate averaging. SOLAR leaves the optimizer update unchanged and controls the LR across parameter groups from observed training states. The AdamW and Muon experiments test whether this scheduler-level control remains useful across distinct optimizer families.

Learning to Optimize. Learning to Optimize (L2O) replaces hand-designed update rules with learned ones (Chen et al., 2022). Learned optimizers map gradients to updates (Andrychowicz et al., 2016; Li & Malik, 2017; Ravi & Larochelle, 2017); hierarchical mod-

els (Wichrowska et al., 2017), large-scale meta-training (Metz et al., 2022), and constrained update spaces (Liu et al., 2023; He et al., 2024) improve their reach. SOLAR learns a lower-dimensional object. The base optimizer retains its update rule, and the policy controls a bounded residual around an external schedule. This restriction makes online acquisition feasible inside the target pretraining run.

3 PRELIMINARIES

Notation. We consider LLM pretraining as minimizing a loss $\mathcal{L}(w)$ over parameters $w \in \mathbb{R}^d$ with stochastic gradients $g_t = \nabla\mathcal{L}(w_t)$. The parameters are partitioned into G controlled groups, each assigned a learning rate $\eta_{t,g} > 0$. SOLAR supplies $\{\eta_{t,g}\}_{g=1}^{G}$ to a standard optimizer (e.g., AdamW, Muon), which internally produces the next iterate w_{t+1} during optimization.

Learning Rate Scheduling as a Sequential Decision Problem. We formulate online LR scheduling as a sequential decision process. At step t, the scheduler observes state $s_t = \{s_{t,g}\}_{g=1}^{G}$, outputs the group-wise LR action vector $a_t = \{a_{t,g}\}_{g=1}^{G}$, and receives rewards $r_{t+1} = \{r_{t+1,g}\}_{g=1}^{G}$. The full action vector is applied to one LLM optimizer step, which jointly determines the next training state and the group-wise rewards.

Proximal Policy Optimization (PPO). We train the scheduler with PPO (Schulman et al., 2017) using a parameter-shared independent update across groups. Each controlled group contributes a likelihood ratio paired with its own advantage. The clipped terms are averaged to update the shared actor–critic. Appendix A.2 gives the exact objective, network architecture, and hyperparameters.

4 SOLAR: STATE-DRIVEN ONLINE LEARNING RATE SCHEDULER

> **SOLAR at a Glance.** A base schedule specifies warmup and decay. SOLAR learns the missing closed-loop correction from the current run. It maps global and parameter-group states to bounded residual actions that re-anchor to the base every step. A progress-aware reward trains this policy online, and a Circuit-Breaker restores the last safe state after a severe loss spike.

4.1 LIGHTWEIGHT STATE REPRESENTATION

SOLAR's state design follows a minimal-variance principle: it relies on a compact set of low-variance optimization signals that are informative yet cheap to compute, avoiding high-dimensional or optimizer-internal statistics. In the dense implementation, each controlled parameter group corresponds to one trainable parameter tensor; the MoE implementations follow the module partition of their respective codebases. SOLAR assigns one learning-rate multiplier per group. To minimize the overhead of online scheduling, SOLAR adopts a lightweight state representation built on a small set of universal optimization signals: loss statistics, gradient norms, and parameter norms.

State representation. For controlled parameter group g, the state is defined as $s_{t,g} = [s_t^{\text{global}};\ s_{t,g}^{\text{local}}]$. The global state summarizes the macro training phase and loss dynamics:

$$s_t^{\text{global}} = [\tau_t,\ \log L_t,\ \nu_t,\ \Delta\text{EMA}_t]\,, \tag{1}$$

where τ_t is the normalized training progress, L_t is the current loss, ν_t measures short-window loss fluctuation, and ΔEMA_t captures the discrepancy between short-term and long-term exponential moving averages of the loss. The local state summarizes controlled-parameter-group-level context:

$$s_{t,g}^{\text{local}} = \left[\log \eta_{t,g}^{\text{base}},\ \log\|\nabla_{t,g}\|,\ a_{t-1,g},\ d_g,\ \log\|w_{t,g}\|,\ \Delta\log\|\nabla_{t,g}\|\right], \tag{2}$$

where $\nabla_{t,g}$ is the current mini-batch gradient, $a_{t-1,g}$ is the previous scheduling action, d_g is the depth index, $w_{t,g}$ is the trainable tensor, and $\Delta\log\|\nabla_{t,g}\|$ measures recent changes in gradient magnitude. This compact representation keeps the scheduler lightweight and decoupled from optimizer-internal statistics. Detailed mathematical formulations for all state features, including moving average decay rates and numerical stability constants, are provided in Appendix A.1.

4.2 Action Space: Stochastic Group-wise Residual Modulation

Given the observed state, a conventional action design is to let the policy directly output an absolute learning rate deterministically. However, accurately predicting a single optimal learning rate at each step is inherently difficult in noisy, large-scale, and highly non-stationary LLM pretraining. SOLAR addresses this challenge through two coupled design choices under practical constraints.

Stochastic group-wise residual actions. Instead of predicting a deterministic scalar learning rate, SOLAR samples a joint collection of one-dimensional bounded residual learning-rate actions across controlled parameter groups from a factorized squashed Gaussian policy. The joint policy factorizes across groups, with one squashed Gaussian factor per controlled parameter group. For each controlled parameter group g, the scheduler outputs a mean $\mu_{t,g}$, while the policy uses a single learnable scalar standard deviation σ shared across all controlled parameter groups. We sample a latent variable, apply the hyperbolic tangent, and clip only the executed action for numerical safety:

$$u_{t,g} \sim \mathcal{N}(\mu_{t,g}, \sigma^2), \qquad \widetilde{a}_{t,g} = \tanh(u_{t,g}), \qquad a_{t,g} = \mathrm{clip}(\widetilde{a}_{t,g}, -1 + 10^{-4}, 1 - 10^{-4}).$$

The stochastic policy explores group-specific corrections, while the bounded action limits the damage of any single decision. We use group-specific means $\mu_{t,g}$ and one learned scalar variance shared across groups; Appendix A.2 gives the implementation.

Residual LR modulation. A base schedule encodes the coarse LR profile. SOLAR keeps that profile outside the learned object and controls a residual:

$$\eta_{t,g} = \eta_{t,g}^{\mathrm{base}} \cdot \exp(\alpha_t a_{t,g}), \tag{3}$$

where $\alpha_t \geq 0$ controls the residual range. We linearly warm up α_t from 0 to α during early training. Because each multiplier is applied to the current base LR, the action cannot recursively redefine later LRs. The policy can focus on state-dependent corrections instead of relearning warmup and decay.

4.3 Reward Design for Sustained Optimization

After action a_t and the optimizer update, the next ordinary forward/backward pass produces L_{t+1} and ∇_{t+1}. SOLAR reuses these detached quantities and adds no language-model forward or backward pass. A one-step loss reward is too noisy to describe sustained progress, so the reward combines immediate improvement, an EMA trend, and group-wise stability:

$$r_{t+1,g} = r_{t+1}^{\mathrm{perf}} + r_{t+1}^{\mathrm{trend}} - p_{t+1,g}^{\mathrm{stab}}, \tag{4}$$

where r_{t+1}^{perf} rewards step-wise loss reduction, r_{t+1}^{trend} encourages improvements over a longer EMA horizon, and $p_{t+1,g}^{\mathrm{stab}}$ is a signed, controlled-group-wise stability shaping term (Appendix A.3). It compares each group's post-update gradient norm with its recent trend, crediting gradients that settle below the trend and penalizing growth above it. The two progress terms remain tied to the global language-modeling objective.

4.4 Safety and Recovery: Circuit-Breaker Mechanism

Rare loss spikes can occur during LLM pretraining (Chowdhery et al., 2023; Zhang et al., 2022; Zeng et al., 2023) and may invalidate online exploration. SOLAR therefore adds a global emergency trigger that terminates unsafe rollouts and restores the latest safe checkpoint. Specifically, we augment the reward with an explicit instability penalty:

$$r_{t+1,g}^{\mathrm{final}} = r_{t+1,g} - \lambda_{\mathrm{cb}} \psi_{t+1}, \tag{5}$$

where λ_{cb} is a large penalty coefficient, and ψ_{t+1} is a binary instability indicator defined as

$$\psi_{t+1} = \mathbb{1}\left[L_{t+1} > \kappa_L \cdot L_{t+1}^{\mathrm{ema}}\right], \tag{6}$$

with κ_L denoting a safety threshold, L_{t+1} the post-update training loss, and L_{t+1}^{ema} its exponential moving average. Unlike the per-group stability shaping term in (4), the Circuit-Breaker acts as a

Algorithm 1: State-Driven Online Learning Rate Scheduler (SOLAR)

Input: Base scheduler $\{\eta_{t,g}^{\text{base}}\}$, optimizer $\mathcal{O}$, policy π_θ, value network V_ϕ, total steps T

for $t = 0, \ldots, T-1$ **do**

// 1. State Construction

Construct scheduler states $\{s_{t,g}\}_{g=1}^{G}$;

// 2. Per-Group LR Action Sampling

Sample $\{u_{t,g}\}_{g=1}^{G}$, form $\{a_{t,g}\}_{g=1}^{G}$, and record $\{\ell_{t,g}^{\text{old}}\}_{g=1}^{G}$;

Set learning rates $\eta_{t,g} = \eta_{t,g}^{\text{base}} \exp(\alpha_t a_{t,g})$ for all g;

// 3. Environment Step

Execute one optimizer step with $\{\eta_{t,g}\}_{g=1}^{G}$;

Compute group-wise final rewards $\{r_{t+1,g}^{\text{final}}\}_{g=1}^{G}$ and instability indicator ψ_{t+1};

Store $(s_t, u_t, a_t, \ell_t^{\text{old}}, r_{t+1}^{\text{final}}, s_{t+1})$, retaining the group axis, in the rollout buffer;

// 4. Policy Update & Safety Net

if *the PPO update condition is satisfied or* $\psi_{t+1} = 1$ **then**

Update π_θ and V_ϕ using PPO;

if $\psi_{t+1} = 1$ **then**

// Circuit-Breaker Triggered

Emit abort signal;

The outer training loop restores model and optimizer states from the latest safe checkpoint;

global emergency signal triggered only by severe loss spikes. When $\psi_{t+1} = 1$, the current rollout is immediately terminated and the scheduler performs a PPO update on the truncated trajectory. The training loop then restores the model and optimizer states from the latest safe checkpoint while retaining the updated PPO state. This prevents post-failure samples from contaminating subsequent policy updates and provides an explicit failure signal that encourages the scheduler to internalize stability constraints.

4.5 Implementation Details and Overhead

SOLAR uses a two-layer MLP actor–critic trained with PPO; Appendix A.2 gives the architecture. Appendix C.14 separates steady-state step time from full-run wall-clock accounting.

5 Experiments

We ask four questions: Does SOLAR improve final pretraining outcomes? Which control choices make online learning effective? Does the residual policy transfer across scales and base schedules? What computational cost does the controller add?

5.1 Main Pretraining Results

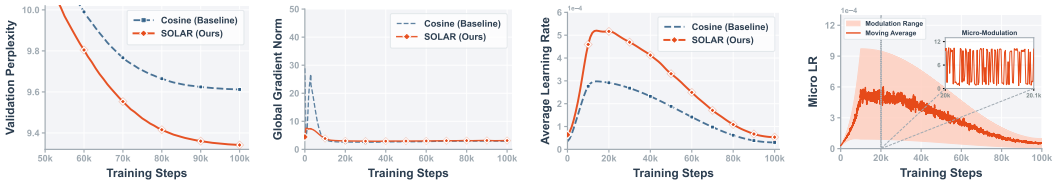
5.1.1 Dense Llama 2 Pretraining across Optimizers

We evaluate on C4 (Raffel et al., 2020) across four Llama 2 (Touvron et al., 2023b) scales (60M to 1B) with AdamW and Muon, following the setup of Zhao et al. (2024). Static schedules, online tuners, and schedule-free baselines share the model, data, optimizer, and training budget. Table 1 reports validation PPL at the final training update. SOLAR-specific hyperparameters are selected once at 60M and then kept fixed. Appendix B.2 gives the complete selection and evaluation protocol.

Both SOLAR modes outperform every non-SOLAR alternative wherever they are evaluated. At 1B, SOLAR-online changes AdamW+Cosine from 16.52 to 14.83 and Muon+Cosine from 14.36 to 13.79. At the larger target scales, SOLAR-frozen further lowers five of the six seed-52 online results by reusing the source-acquired controller. Blockwise LR does not consistently improve the tuned base, while AutoLRS and MECHANIC trail the static schedules in most settings. Schedule-Free AdamW (Defazio et al., 2024) improves slightly over Cosine at all four scales and is the strongest non-SOLAR alternative at 350M, yet remains above SOLAR. The gains persist across the repeated runs summarized in Appendix C.2.

Table 1: Seed-52 final-checkpoint validation perplexity on C4. SOLAR-online acquires its policy within the target run; SOLAR-frozen reuses a policy acquired from full-length online runs at 60M, with no target PPO updates. AvgLR Replay replays SOLAR's step-wise mean LR. † marks official AdamW-based implementations. Multi-seed results appear in Appendix C.2.

	Model Scale (Validation Perplexity ↓)			
Method	**60M**	**130M**	**350M**	**1B**
Base Optimizer: AdamW				
Cosine	30.49	24.52	18.31	16.52
AvgLR Replay	30.68	25.61	21.69	20.06
WSD	29.80	23.96	18.75	16.29
CLR	31.78	28.14	21.43	20.22
†Blockwise LR	31.06	24.38	18.56	16.74
AutoLRS	31.46	25.02	20.07	19.08
MECHANIC	32.14	26.95	20.60	19.25
†Prodigy	45.27	27.51	22.14	20.75
†Schedule-Free	30.28	24.49	18.14	16.45
SOLAR (online)	**28.92**	**22.79**	**17.23**	**14.83**
SOLAR (frozen)	–	**22.41**	**17.37**	**14.63**
Base Optimizer: Muon				
Cosine	29.20	22.55	16.87	14.36
AvgLR Replay	30.23	24.41	18.99	17.07
WSD	29.18	22.47	16.62	14.31
CLR	30.79	26.15	19.19	18.62
AutoLRS	30.07	23.63	18.95	16.46
MECHANIC	31.04	25.52	18.88	16.68
SOLAR (online)	**28.63**	**21.99**	**16.35**	**13.79**
SOLAR (frozen)	–	**21.87**	**16.22**	**13.54**



(a) Validation PPL (b) Global Grad Norm (c) Macro: Average LR (d) Micro: Attention LR

Figure 3: Qwen2-MoE 1B pretraining on The Pile. SOLAR improves validation perplexity, suppresses early gradient-norm spikes, and maintains a higher average LR through group-wise micro-modulation. Panel (d) shows one representative attention projection group; the solid line is a 200-step moving average and the shaded region visualizes the corresponding step-wise variation.

5.1.2 MoE Pretraining under Non-Stationary Dynamics

MoE training introduces routing-induced stochasticity and sparse, uneven gradient updates, which often require conservative optimization settings (Fedus et al., 2022; Zoph et al., 2022). We evaluate SOLAR on Qwen2-MoE 1B (Yang et al., 2024) pretrained on The Pile under the setup in Appendix B.4. Figure 3 shows a final PPL improvement from 9.61 to 9.34. SOLAR uses a higher average LR while producing a lower, smoother gradient-norm trace than Cosine in this run. At the module level (Figure 3d), the effective LRs fluctuate around a decaying profile, with distinct actions across groups. Section 5.3 examines these traces. Appendix C.12 reports the separate 3B DeepSeek-V2-style MoE run in Megatron, where final PPL changes from 10.73 to 10.38.

Runtime and Circuit-Breaker usage. No main-table or MoE run triggers rollback. On 1B AdamW, online SOLAR adds 1.23% to full-run wall-clock time and 1.27% to steady-state step time; the frozen

mode adds 0.76% and 0.81%, respectively. Appendix C.14 defines both measurements and reports the remaining settings.

5.2 Frozen Residual-Policy Transfer Across Model Scales

The residual parameterization expresses actions relative to the current base LR. We test whether this dimensionless policy can be acquired on a small proxy and reused at larger dense scales.

Setup. We train SOLAR on five 60M pretraining trajectories of 11K updates. The policy is then frozen and applied to 130M, 350M, and 1B models within the same optimizer family. Each target run combines the frozen policy with its base schedule, performs no PPO updates, and does not retune SOLAR-specific hyperparameters. All source-policy parameters, including the final learned shared $\log \sigma$, remain fixed, while group-wise actions are recomputed from the current target state at every update.

Results. SOLAR-frozen improves over the corresponding target-tuned Cosine and WSD schedules at every larger scale in Table 1. It yields the lower PPL in five of six matched seed-52 comparisons with SOLAR-online, while online acquisition yields the lower PPL under the corpus and optimizer shifts in Appendix C.3. Across repeated 130M runs, SOLAR-online and SOLAR-frozen reach mean final PPLs of 22.86 and 22.76 with AdamW, and 22.05 and 22.09 with Muon, respectively. Frozen reuse improves from 25.04 PPL at random initialization to 22.41 after five full-length 60M source runs (Appendix C.3). Appendix C.5 tests the same policy under $0.5\times$, $1\times$, and $2\times$ target base LRs: it beats the matched Cosine run in all six settings and completes both $2\times$ runs where Cosine diverges. A separate μP width-transfer experiment removes target-scale base-LR search and retains the improvement, while a C4-to-Pile experiment tests simultaneous scale and corpus shift.

5.3 Mechanistic Evidence

Unless noted otherwise, we analyze the seed-52 130M Llama 2 run to characterize SOLAR's structured control. Figure 4 shows that SOLAR acts along three axes: temporal adaptation, module-wise redistribution, and module-specific feedback to gradient statistics.

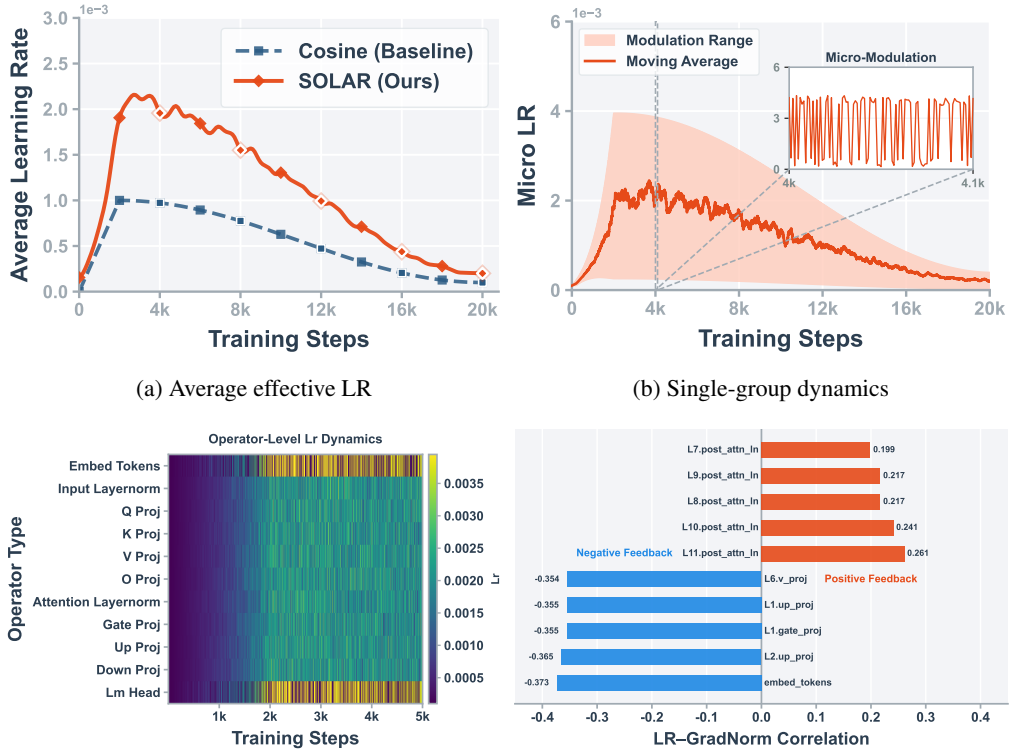
Control structure. Matched controls identify which design choices separate improvement from degradation. A recursive global PPO controller reaches 27.09 final PPL. Re-anchoring and bounding the same global interface lowers PPL to 23.74; replacing the global action with group-wise control reaches 22.87 ($n = 2$, seeds 42 and 52). Group-wise hypergradient control reaches 24.26, while an LLM adaptation of the layer-wise GANNO controller reaches 24.95. Untrained, near-deterministic, and state-blind SOLAR policies remain above 25.0. The anchored, state-conditioned global interface reaches 23.74, and group-wise control provides a further 0.87-PPL gain. Appendix C.4 gives the implementations, tuning protocol, and per-seed results.

Temporal adaptation. SOLAR maintains a higher average LR than Cosine, while AvgLR Replay ends at 25.61 PPL (Figure 4a). A normalized static group profile reaches 24.19 PPL, compared with 22.79 for SOLAR-online on the same seed. These controls show that neither a scalar temporal schedule nor a static group allocation reproduces SOLAR's online trajectory. Appendix C.6 gives the protocols and results.

Module-wise redistribution. Figure 4c shows a stable module-level allocation pattern: vocabulary-facing components (token embeddings, LM head) receive larger multipliers, while internal attention and MLP projections are more constrained. This structure is consistent across training and suggests that SOLAR reallocates optimization capacity toward modules that appear more sensitive to learning-rate variation. Such structural allocation cannot be reproduced by a scalar global schedule.

Module-specific feedback. SOLAR learns differentiated LR–GradNorm relationships (Figure 4d). Dense transformation matrices exhibit negative correlation, while several post-attention LayerNorm parameters show positive correlation. The mapping therefore changes across modules rather than applying a uniform inverse function of gradient norm.

Appendix C provides state, action, reward, Circuit-Breaker, and base-LR analyses. Appendix C.4 contains the matched learned-controller comparisons.



(a) Average effective LR

(b) Single-group dynamics

(c) Module-wise allocation

(d) LR–GradNorm relation

Figure 4: Mechanistic analysis of SOLAR on 130M Llama 2. (a)–(d): average effective LR, single-group LR dynamics, module-wise LR allocation, and per-module LR–GradNorm correlation, respectively. In (b), the solid line is a 200-step moving average and the shaded region visualizes the corresponding step-wise variation.

6 Conclusion and Limitations

We presented SOLAR, an online LR scheduler that uses a base schedule for the coarse LR profile while adapting to the realized optimization state. Its policy learns bounded, state-dependent group-wise corrections that re-anchor to the base at every step, guided by lightweight training features and a progress-aware reward. Across dense and MoE settings, SOLAR improves final PPL over tuned static schedules and automatic LR tuners under both AdamW and Muon, with repeated gains through 1B. On matched seeds, controls show that base anchoring and action bounds improve a global PPO controller from 27.09 to 23.74 PPL, while group-wise control reaches 22.87.

The controls also clarify how the design divides the scheduling problem. Re-anchoring preserves the base scheduler's warmup–decay profile, bounded residual actions limit the effect of individual policy decisions, and group-wise feedback allows the correction to differ across parameter tensors as the run evolves. This structure supports two complementary operating modes. SOLAR-online learns the controller within the current pretraining run, while SOLAR-frozen reuses an acquired policy without target-side PPO updates or SOLAR-specific retuning. A policy trained on a 60M proxy transfers to larger dense scales and beats matched Cosine runs across the tested fourfold base-LR range. The cross-corpus, reward, and μP results in the appendix extend the evaluation across changes in data, reward composition, and parameterization.

These results provide a practical route to adaptive LR control in LLM pretraining. The primary limitation of SOLAR is that its validation is still limited to dense models up to 1B parameters and a supplementary 3B MoE setting, so its behavior at larger scales remains untested. We leave this investigation to future work.

AI USE STATEMENT

Generative AI tools were used only to refine wording and proofread the manuscript for clarity. The authors produced the technical contributions, conducted the experiments and analyses, and verified all reported results and conclusions.

REPRODUCIBILITY STATEMENT

Appendix A specifies SOLAR's state, action, reward, PPO update, and Circuit-Breaker mechanism. Appendix B documents the model, data, optimizer, scheduler, hardware, and hyperparameter configurations, including the final-PPL protocol. Appendix C reports repeated runs, ablations, transfer evaluations, and matched controls supporting the empirical results.

REFERENCES

Marcin Andrychowicz et al. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with Hypergradient descent. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=BkrsAzWAb.

Tianlong Chen et al. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research (JMLR)*, 2022.

Xiangning Chen et al. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Aakanksha Chowdhery et al. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. URL https://www.jmlr.org/papers/v24/22-1144.html.

Ashok Cutkosky, Aaron Defazio, and Harsh Mehta. Mechanic: A learning rate tuner. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Christian Daniel, Jonathan Taylor, and Sebastian Nowozin. Learning step size controllers for robust neural network training. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2016. URL https://ojs.aaai.org/index.php/AAAI/article/view/10187.

DeepSeek-AI et al. DeepSeek-v2: A strong, economical, and efficient Mixture-of-Experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by D-Adaptation. In *International Conference on Machine Learning (ICML)*, 2023.

Aaron Defazio, Xingyu Alice Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. doi: 10.52202/079017-0320. URL https://papers.nips.cc/paper_files/paper/2024/hash/136b9a13861308c8948cd308ccd02658-Abstract-Conference.html.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 2011. URL https://jmlr.org/papers/v12/duchi11a.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research (JMLR)*, 2022.

Leo Gao et al. The Pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Samantha Hansen. Using deep q-learning to control optimization hyperparameters, 2016. URL https://arxiv.org/abs/1602.04062.

Yutong He, Qiulin Shang, Xinmeng Huang, Jialin Liu, and Kun Yuan. A mathematics-inspired learning-to-optimize framework for decentralized optimization, 2024. URL https://arxiv.org/abs/2410.01700.

Shengding Hu et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

Tianjin Huang, Ziquan Zhu, Gaojie Jin, Lu Liu, Zhangyang Wang, and Shiwei Liu. SPAM: Spike-aware Adam with momentum reset for stable LLM training. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7a70ad3d9c704fb9b81b5c69eda722dc-Abstract-Conference.html.

Max Jaderberg et al. Population based training of neural networks, 2017. URL https://arxiv.org/abs/1711.09846.

Yuchen Jin et al. AutoLRS: Automatic learning-rate schedule by Bayesian optimization on the fly. In *International Conference on Learning Representations (ICLR)*, 2021.

Keller Jordan et al. Muon: An optimizer for hidden layers in neural networks. Blog post, 2024. URL https://kellerjordan.github.io/posts/muon/. Accessed: 2026-04-22.

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

Ke Li and Jitendra Malik. Learning to optimize. In *International Conference on Learning Representations (ICLR)*, 2017.

Jialin Liu, Xiaohan Chen, Zhangyang Wang, Wotao Yin, and HanQin Cai. Towards constituting mathematical structures for learning to optimize. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21426–21449, 2023. URL https://proceedings.mlr.press/v202/liu23e.html.

Jingyuan Liu et al. Muon is scalable for LLM training, 2025. URL https://arxiv.org/abs/2502.16982.

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

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

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.

Luke Metz, James Harrison, C. Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, and Jascha Sohl-Dickstein. VeLO: Training versatile learned optimizers by scaling up, 2022. URL https://arxiv.org/abs/2211.09760.

Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. In *International Conference on Machine Learning (ICML)*, 2024.

Colin Raffel et al. Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research (JMLR)*, 2020.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning (ICML)*, 2018.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism, 2019. URL https://arxiv.org/abs/1909.08053.

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

Leslie N. Smith and Nicholay Topin. Super-Convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 369–386. SPIE, 2019. doi: 10.1117/12.2520589. URL https://doi.org/10.1117/12.2520589.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Shreyas Subramanian, Vignesh Ganapathiraman, and Aly El Gamal. Learned learning rate schedules for deep neural network training using reinforcement learning. In Krystal Maughan, Rosanne Liu, and Thomas F. Burns (eds.), *The First Tiny Papers Track at ICLR 2023*, 2023.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. Spike no more: Stabilizing the Pre-training of large language models. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2312.16903.

Team OLMo et al. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Kale-ab Tessera, Callum Rhys Tilbury, Sasha Abramowitz, Ruan de Kock, Omayma Mahjoub, Benjamin Rosman, Sara Hooker, and Arnu Pretorius. Generalisable agents for neural network optimisation, 2023. URL https://arxiv.org/abs/2311.18598.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. URL https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf.

Hugo Touvron et al. LLaMA: Open and efficient foundation language models, 2023a. URL https://arxiv.org/abs/2302.13971.

Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models, 2023b. URL https://arxiv.org/abs/2307.09288.

Jinbo Wang, Mingze Wang, Zhanpeng Zhou, Junchi Yan, Weinan E, and Lei Wu. The Sharpness disparity principle in Transformers for accelerating language model Pre-Training. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64859–64879, 2025. URL https://proceedings.mlr.press/v267/wang25dl.html.

Olga Wichrowska et al. Learned optimizers that scale and generalize. In *International Conference on Machine Learning (ICML)*, 2017.

Yuanhao Xiong, Li-Cheng Lan, Xiangning Chen, Ruochen Wang, and Cho-Jui Hsieh. Learning to schedule learning rate with graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=k7efTb0un9z.

Chang Xu, Tao Qin, Gang Wang, and Tie-Yan Liu. Reinforcement learning for learning rate control, 2017. URL https://arxiv.org/abs/1705.11159.

Zhen Xu, Andrew M. Dai, Jonas Kemp, and Luke Metz. Learning an adaptive learning rate schedule, 2019. URL https://arxiv.org/abs/1909.09712.

An Yang et al. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Greg Yang et al. Tensor programs v: Tuning large neural networks via Zero-Shot hyperparameter transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Aohan Zeng et al. GLM-130b: An open bilingual pre-trained model. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=-Aw0rrrPUF.

Susan Zhang et al. OPT: Open pre-trained Transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *International Conference on Machine Learning (ICML)*, 2024.

Shuchen Zhu et al. Accelerating LLM Pre-Training through Flat-Direction dynamics enhancement, 2026. URL https://arxiv.org/abs/2602.22681.

Barret Zoph et al. ST-MoE: Designing stable and transferable sparse expert models, 2022. URL https://arxiv.org/abs/2202.08906.

Appendix

A SOLAR Implementation Details

In this section, we provide the complete implementation details of the SOLAR scheduler, including state construction, action parameterization, PPO training configurations, and the automated Circuit-Breaker mechanism.

A.1 State Definition and Feature Extraction

This appendix expands the compact state definition in Section 4.1 by specifying normalization constants, EMA coefficients, window sizes, and missing-gradient handling.

State definition. For controlled parameter group g, the scheduler state is defined as the concatenation of global optimization features and local group-specific features:

$$s_{t,g} = \left[s_t^{\text{global}};\ s_{t,g}^{\text{local}} \right].$$

Global features. The global state captures the overall training dynamics:

$$s_t^{\text{global}} = [\tau_t,\ \log L_t,\ \nu_t,\ \Delta\text{EMA}_t].$$

In implementation, the normalized training progress is $\tau_t = t/T$, where T is the total number of training steps. The loss term is implemented as:

$$\log L_t = \log(L_t + \epsilon),$$

where $\epsilon = 10^{-8}$ is the numerical stability constant (used uniformly across all log and EMA denominator computations in this appendix). The short-window loss fluctuation is computed from a sliding window of recent losses:

$$\nu_t = \frac{\text{Std}(\mathcal{H}_t)}{L_t + \epsilon},$$

where $\mathcal{H}_t$ stores the most recent $w = 20$ losses; when fewer than 20 steps have elapsed, we use all available samples. To characterize short-term versus long-term optimization trends, we maintain two exponential moving averages (EMA) of the loss with $\beta_s = 0.9$ and $\beta_\ell = 0.99$, initialized as $\text{EMA}_0^{\text{short}} = \text{EMA}_0^{\text{long}} = L_0$:

$$\text{EMA}_t^{\text{short}} = \beta_s \text{EMA}_{t-1}^{\text{short}} + (1 - \beta_s) L_t, \tag{7}$$

$$\text{EMA}_t^{\text{long}} = \beta_\ell \text{EMA}_{t-1}^{\text{long}} + (1 - \beta_\ell) L_t. \tag{8}$$

The trend difference is then defined as:

$$\Delta\text{EMA}_t = \frac{\text{EMA}_t^{\text{short}} - \text{EMA}_t^{\text{long}}}{\text{EMA}_t^{\text{long}} + \epsilon}.$$

Local features. The local state captures tensor-specific statistics:

$$s_{t,g}^{\text{local}} = \left[\log \eta_{t,g}^{\text{base}},\ \log \|\nabla_{t,g}\|,\ a_{t-1,g},\ d_g,\ \log \|w_{t,g}\|,\ \Delta \log \|\nabla_{t,g}\| \right].$$

The base learning-rate context is $\log \eta_{t,g}^{\text{base}} = \log(\eta_{t,g}^{\text{base}} + \epsilon)$ and the parameter-scale feature is $\log \|w_{t,g}\| = \log(\|w_{t,g}\|_2 + \epsilon)$, both using the same $\epsilon = 10^{-8}$. The gradient-magnitude feature and its recent change are computed respectively as:

$$\log \|\nabla_{t,g}\| = \log(\|\nabla_{t,g}\|_2 + \epsilon), \tag{9}$$

$$\Delta \log \|\nabla_{t,g}\| = \log(\|\nabla_{t,g}\|_2 + \epsilon) - \log(\|\nabla_{t-1,g}\|_2 + \epsilon). \tag{10}$$

The previous action $a_{t-1,g}$ is the action applied to controlled group g at the previous step. The depth feature d_g records the normalized network depth of the module containing controlled group g. If a controlled parameter group has no gradient at step t, we set gradient-dependent local entries ($\log \|\nabla_{t,g}\|$, $\Delta \log \|\nabla_{t,g}\|$) to zero and keep non-gradient entries ($\log \eta_{t,g}^{\text{base}}$, $a_{t-1,g}$, d_g, $\log \|w_{t,g}\|$) unchanged when available. Each state feature is normalized with EMA running statistics using decay 0.99 before it is passed to the policy.

A.2 Action Space, Architecture, and Distributed Execution

Action Parameterization. SOLAR controls hyperparameters at the fine-grained level of controlled parameter groups. Each controlled group corresponds to one trainable parameter tensor in the dense implementation; the MoE implementations use the module partition defined by their respective codebases. For every group, the actor samples a pre-Tanh variable, forms a squashed action, and clips only the value sent to the optimizer:

$$\begin{aligned} u_{t,g} &\sim \mathcal{N}(\mu_{t,g}, \sigma^2), \\ \widetilde{a}_{t,g} &= \tanh(u_{t,g}), \\ a_{t,g} &= \mathrm{clip}(\widetilde{a}_{t,g}, -1 + 10^{-4}, 1 - 10^{-4}). \end{aligned} \tag{11}$$

The PPO log-probability is evaluated from the stored $u_{t,g}$ with the Tanh change-of-variables correction, before the numerical clipping that produces $a_{t,g}$.

The scheduler applies multiplicative residuals to the underlying base scheduler. The executed learning rate for controlled parameter group g at step t is:

$$\eta_{t,g} = \eta_{t,g}^{\mathrm{base}} \cdot \exp\left(\alpha_t a_{t,g}\right), \tag{12}$$

where $\eta_{t,g}^{\mathrm{base}}$ is the base learning rate and α_t is the action scale. To ensure stability during the initial phase, we linearly warm up the action scale: $\alpha_t = \alpha \cdot (t/T_{\mathrm{warmup}})$ for $t < T_{\mathrm{warmup}}$, and $\alpha_t = \alpha$ afterward, where $T_{\mathrm{warmup}} = 0.1T$ across all experiments. We set the maximum action scale to $\alpha = 1.3$.

Architecture and PPO Setup. We instantiate SOLAR with a lightweight actor-critic network. Given the state representation $s_{t,g}$ for controlled parameter group g, a shared MLP encoder first produces a hidden representation:

$$h_{t,g} = f_{\mathrm{enc}}(s_{t,g}),$$

where f_{enc} is a two-layer MLP with input dimension 10 (matching the state dimension), hidden dimension 256, and Tanh activations; no layer normalization is used. This representation is then passed to separate actor and critic heads, which are implemented as linear projections:

$$\mu_{t,g} = W_\mu h_{t,g} + b_\mu, \qquad V_\phi(s_{t,g}) = W_v h_{t,g} + b_v.$$

The actor policy $\pi_\theta(\widetilde{a}_{t,g} \mid s_{t,g})$ is a one-dimensional Tanh-Normal distribution with mean $\mu_{t,g}$ and a globally learnable scalar log-standard-deviation parameter $\log\sigma$ shared across all controlled parameter groups.

The squashed action distribution factorizes across controlled groups,

$$\pi_\theta(\widetilde{a}_t \mid s_t) = \prod_{g=1}^{G} \pi_\theta(\widetilde{a}_{t,g} \mid s_{t,g}), \tag{13}$$

where all factors share the actor parameters θ. For the stored pre-Tanh sample $u_{t,g}$, define

$$\ell_{t,g}(\theta) = \log\mathcal{N}\big(u_{t,g}; \mu_\theta(s_{t,g}), \sigma^2\big) - \log\big(1 - \tanh^2(u_{t,g})\big), \tag{14}$$

which is the Tanh-Normal log-probability evaluated from the stored $u_{t,g}$. All action factors share θ and act on the same LLM transition. Policy learning uses a parameter-shared independent update. Let θ_{old} denote the behavior-policy parameters used to collect the rollout. For group g, the other action factors remain at their behavior-policy samples while its local objective is evaluated under θ:

$$J_g^{\mathrm{ind}}(\theta; \theta_{\mathrm{old}}) = \mathbb{E}_{\substack{\widetilde{a}_{\tau,g} \sim \pi_\theta(\cdot \mid s_{\tau,g}) \\ \widetilde{a}_{\tau,-g} \sim \pi_{\theta_{\mathrm{old}}}(\cdot \mid s_{\tau,-g})}} \left[\sum_{\tau=t}^{T-1} \gamma^{\tau-t} r_{\tau+1,g}^{\mathrm{final}} \right]. \tag{15}$$

Using group-wise GAE, the independent policy update uses the gradient estimate

$$\widehat{g}_g := \widehat{\mathbb{E}}_t \Big[\hat{A}_{t,g} \, \nabla_\theta \ell_{t,g}(\theta)\big|_{\theta=\theta_{\mathrm{old}}} \Big]. \tag{16}$$

PPO applies likelihood-ratio clipping to each group objective. The group-specific ratio is

$$\rho_{t,g}(\theta) = \exp(\ell_{t,g}(\theta) - \ell_{t,g}(\theta_{\mathrm{old}})). \tag{17}$$

The per-group log-probabilities are not summed when computing $\rho_{t,g}$. Averaging the group surrogates gives

$$\mathcal{L}_{\mathrm{PPO}}(\theta) = \mathbb{E}_t\left[\frac{1}{G}\sum_{g=1}^{G}\min\Big(\rho_{t,g}(\theta)\hat{A}_{t,g}, \mathrm{clip}(\rho_{t,g}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}})\,\hat{A}_{t,g}\Big)\right], \tag{18}$$

where $\hat{A}_{t,g}$ is the generalized advantage estimate (GAE) (Schulman et al., 2016) for group g, and ϵ_{clip} is the clipping threshold. The minimum and clipping operations are applied to each (t, g) entry before the shared actor receives their average. The value and entropy terms use the same group-wise reduction:

$$\mathcal{L}_V(\phi) = \mathbb{E}_t\left[\frac{1}{G}\sum_{g=1}^{G}\Big(V_\phi(s_{t,g}) - \hat{R}_{t,g}\Big)^2\right], \tag{19}$$

$$\mathcal{L}_H(\theta) = \mathbb{E}_t\left[\frac{1}{G}\sum_{g=1}^{G}\mathcal{H}[\pi_\theta(\cdot \mid s_{t,g})]\right]. \tag{20}$$

The actor–critic minimizes

$$\mathcal{J}(\theta, \phi) = -\mathcal{L}_{\mathrm{PPO}}(\theta) + \lambda_v\mathcal{L}_V(\phi) - \lambda_e\mathcal{L}_H(\theta), \tag{21}$$

where $\hat{R}_{t,g}$ is the target return, λ_v is the value-loss coefficient, and λ_e is the entropy coefficient.

In our implementation, we optimize $\mathcal{J}(\theta, \phi)$ using the Adam optimizer with a learning rate of 3×10^{-4}. Standard PPO hyperparameters include a discount factor $\gamma = 0.99$, a clipping threshold $\epsilon_{\mathrm{clip}} = 0.2$, a value loss coefficient $\lambda_v = 0.5$, and an entropy coefficient $\lambda_e = 0.05$. During normal training, the RL agent collects trajectories and performs an update every 50 steps, with $K = 4$ optimization epochs per update.

For a rollout of T_r decision steps, states have shape $[T_r, G, d_s]$; pre-Tanh samples, executed actions, log-probabilities, ratios, returns, values, and advantages have shape $[T_r, G]$. GAE is computed independently along the temporal axis for each controlled group with $\lambda_{\mathrm{GAE}} = 0.95$. Advantages are normalized once over the complete $[T_r, G]$ block as $\texttt{adv} \leftarrow (\texttt{adv} - \texttt{mean(adv)})/(\texttt{std(adv)} + 10^{-7})$ and reused across PPO epochs. Reward normalization and value-loss clipping are not used.

The actor-critic network weights are initialized as: all feature and hidden linear layers use orthogonal initialization with gain $\sqrt{2}$, biases set to 0; the actor output layer uses constant initialization of 0.01 on weights with zero bias, so that the initial mean $\mu_{t,g} \approx 0$; the critic output layer uses orthogonal initialization with gain $\sqrt{2}$ and zero bias; the log-standard-deviation is initialized to 0 (yielding $\sigma = 1.0$) and is learned. Because the scheduler only predicts low-dimensional multiplicative LR corrections while leaving the base optimizer's internal mechanics unchanged, SOLAR remains highly lightweight and seamlessly integrates into standard pretraining pipelines.

Base Scheduler and Transfer Configuration. For the reported dense experiments, the base scheduler uses the Cosine peak LR listed for the corresponding model scale and optimizer in Appendix B.2. The Qwen2-MoE experiment also uses a Cosine base, while the supplementary 3B MoE experiment uses WSD as specified in Appendix C.12. In frozen residual-policy transfer, the target combines its corresponding base schedule with the source policy. The policy performs no target-scale PPO updates; the source policy's final learned parameters, including the shared $\log\sigma$, are transferred and kept fixed together with α, action warmup, reward coefficients, and PPO settings.

Distributed Execution. To minimize synchronization overhead during large-scale pretraining, the RL agent operates exclusively on the rank-0 worker. The rank-0 process evaluates the policy, samples actions, and updates the PPO agent. The sampled actions are then broadcast to all other workers to ensure identical optimizer states across the distributed cluster. In the Distributed Data Parallel (DDP) setting, gradient norms are computed locally on each rank from the already-synchronized gradients (all-reduced during `backward()`); per-parameter norms are then broadcast from rank-0 rather than all-reduced, since the gradient values are already global and computing norms is a secondary reduction. The full procedure is summarized in Algorithm 2.

A.3 Reward Design and Circuit-Breaker Mechanism

Let L_t and ∇_t denote the loss and synchronized gradient from the forward/backward *before* the optimizer update at step t. The scheduler observes state s_t, samples action a_t, and the optimizer updates parameters. After the update, the next forward/backward produces L_{t+1} and post-update gradient ∇_{t+1}. The reward credited to a_t is then computed as follows. Since the scheduler outputs controlled-parameter-group-wise learning-rate adjustments, each controlled parameter group receives a reward that combines global progress feedback with a controlled-group-wise stability shaping term:

$$r_{t+1,g} = r_{t+1}^{\text{perf}} + r_{t+1}^{\text{trend}} - p_{t+1,g}^{\text{stab}}. \tag{22}$$

Performance term. We first measure the relative improvement over the previous step:

$$\chi_{t+1} = \frac{L_t}{L_{t+1} + 10^{-10}}. \tag{23}$$

The performance term is defined as:

$$r_{t+1}^{\text{perf}} = 20 \cdot \log(\chi_{t+1}). \tag{24}$$

Trend term. We incorporate a smoother long-term signal based on the long EMA of the loss:

$$r_{t+1}^{\text{trend}} = 2 \cdot \frac{\text{EMA}_{t+1}^{\text{long}} - L_{t+1}}{\text{EMA}_{t+1}^{\text{long}} + 10^{-8}}. \tag{25}$$

Controlled-group-wise stability shaping term. We compute stability feedback at the same granularity as the scheduling actions. For each controlled parameter group g, let $\|\nabla_{t+1,g}\|_2$ denote its gradient norm from the backward *after* the optimizer update, and let $m_{t+1,g}$ be its exponential moving average (EMA):

$$m_{t+1,g} = \beta m_{t,g} + (1-\beta)\|\nabla_{t+1,g}\|_2,$$

where $\beta = 0.99$ is the smoothing coefficient. The EMA is initialized from the first observed gradient norm, $m_{0,g} = \|\nabla_{0,g}\|_2$. We define the relative gradient-growth ratio as

$$q_{t+1,g} = \frac{\|\nabla_{t+1,g}\|_2}{m_{t+1,g} + \epsilon}, \tag{26}$$

where $\epsilon = 10^{-8}$ is the numerical stability constant. The controlled-group-wise stability shaping term is

$$p_{t+1,g}^{\text{stab}} = \begin{cases} q_{t+1,g} - 1, & q_{t+1,g} \le \tau_{\text{sev}}, \\ q_{t+1,g} - 1 + \delta_{\text{sev}}, & q_{t+1,g} > \tau_{\text{sev}}, \end{cases} \tag{27}$$

where $\tau_{\text{sev}} = 3.0$ and $\delta_{\text{sev}} = 20$.

This signed term compares each group's post-update gradient norm with its own recent trend. Since it is subtracted from the reward, $q_{t+1,g} < 1$ provides positive shaping, whereas $q_{t+1,g} > 1$ reduces the reward; the severe offset applies when $q_{t+1,g} > \tau_{\text{sev}}$. Because the signal is computed after the update and indexed by controlled group, it provides group-resolved feedback aligned with the group-wise actions. The two progress terms remain shared across groups and tied to the global language-modeling objective. Since the current gradient norm also enters $m_{t+1,g}$, $0 \le q_{t+1,g} \le 1/(1-\beta) = 100$ and $-1 \le p_{t+1,g}^{\text{stab}} \le 119$; no additional clipping is used.

Circuit-Breaker and Restart Procedure. To prevent catastrophic divergence during online exploration and ensure training stability, we implement an automated Circuit-Breaker mechanism. Unlike the controlled-group-wise stability shaping term, which provides dense feedback during normal optimization, the Circuit-Breaker is a global emergency mechanism triggered by severe loss spikes. At each step, we compute the loss spike ratio:

$$\kappa_{t+1} = \frac{L_{t+1}}{L_{t+1}^{\text{ema}} + 10^{-8}}. \tag{28}$$

If $\kappa_{t+1} > \kappa_L$, the Circuit-Breaker is triggered, where $\kappa_L = 1.5$. The optimizer wrapper executes the following procedure:

1. **Punishment:** Applies the global Circuit-Breaker penalty $r^{\text{final}}_{t+1,g} = r_{t+1,g} - \lambda_{\text{cb}}$ for all controlled parameter groups g, with $\lambda_{\text{cb}} = 100$.

2. **Forced Update:** Rank 0 immediately performs a PPO update using the transitions accumulated so far, ensuring the agent promptly learns from this catastrophic failure event.

3. **Abort Signal:** Sets an `abort_training=True` flag and broadcasts it to all workers.

The optimizer wrapper does not restore model weights. It returns an abort signal to the outer training loop, which truncates the corrupted trajectory and restores the latest safe model and optimizer state. The forced PPO update is retained rather than replaced by the checkpointed PPO state.

Hyperparameter settings. The numerical constants in (22)–(28) are fixed design choices with distinct meanings. The coefficient 20 in r^{perf}_{t+1} scales the stepwise improvement signal, and the coefficient 2 in r^{trend}_{t+1} scales the long-term trend signal. The stability shaping term uses $\tau_{\text{sev}} = 3.0$ and adds $\delta_{\text{sev}} = 20$ when gradient growth crosses this threshold. The Circuit-Breaker uses $\kappa_L = 1.5$ as the loss-spike trigger and $\lambda_{\text{cb}} = 100$ as the emergency penalty. In our main training runs, the Circuit-Breaker is not triggered, so κ_L and λ_{cb} do not affect the nominal optimization trajectory. We selected this configuration once on a small proxy setting and kept it fixed across all experiments.

A.3.1 Controller Coefficient Census and Reward Leave-One-Out Analysis

The calibration covers four reward coefficients and the short-loss EMA decay used in the state. Table 2 lists these coefficients together with the fixed state-normalization decay. Sweeping one coefficient at a time produces final PPL between 22.80 and 22.85, compared with 22.79 at the default point. The largest deviation, 0.06 PPL, is below the observed 130M SOLAR seed standard deviation of 0.08.

Table 2: Controller-coefficient census on 130M/C4/AdamW.

Quantity	Default	Status	Values evaluated
Short-term progress scale	20	tunable	$\{10, 20, 40\}$ and removal
EMA trend scale	2	tunable	$\{1, 2, 4\}$ and removal
Severe stability offset	20	tunable	$\{10, 20, 40\}$ and removal
Severe stability threshold	3.0	tunable	$\{2.5, 3.0, 4.0\}$
State short-loss EMA decay β_s	0.90	tunable	$\{0.90, 0.95\}$
State-normalization EMA decay	0.99	fixed	fixed across experiments

The coefficient sweep tests local calibration, while the leave-one-out experiment measures the contribution of each reward component. Removing immediate progress, trend, and stability increases final PPL by 1.94, 0.90, and 0.30, respectively. Removing stability also causes the only Circuit-Breaker activation observed under nominal hyperparameters.

Table 3: Reward leave-one-out on 130M/C4/AdamW using seeds 42 and 52.

Variant	Seed 52	Seed 42	Mean PPL	Δ vs. full	CB activations
Full reward	22.79	22.95	22.87	0.00	0/2
− immediate progress	24.91	24.71	24.81	+1.94	0/2
− EMA trend	23.71	23.83	23.77	+0.90	0/2
− stability shaping term	23.24	23.09	23.17	+0.30	1/2

The activation occurs in the no-stability seed-42 run at update 12,340. The Circuit-Breaker restores the latest safe state, after which the run completes. Removing stability raises mean PPL by 0.30 and produces the only recovery event observed under the nominal settings.

Algorithm 2: SOLAR optimizer step (PyTorch-style pseudocode)

Input: Model parameters $\{w_g\}_{g=1}^{G}$, current loss L_t, base scheduler f_{base}, policy π_θ

// From the forward/backward on the current batch

$L_t \leftarrow \text{BROADCASTFROMRANK0}(L_t)$;

Update loss history and short/long EMAs;

Compute global state s_t^{global};

for $g = 1$ **to** G **do**

  Compute local features from $\nabla_{t,g}$, $w_{t,g}$, $\eta_{t,g}^{\text{base}}$, $a_{t-1,g}$, and d_g;

  Form state $s_{t,g}$;

end

if rank $== 0$ **then**

  $\{u_{t,g}, a_{t,g}, \ell_{t,g}^{\text{old}}\}_{g=1}^{G} \leftarrow \pi_\theta.\text{ACT}(\{s_{t,g}\}_{g=1}^{G})$;

else

  Allocate placeholder actions;

end

$\{a_{t,g}\}_{g=1}^{G} \leftarrow \text{BROADCASTFROMRANK0}(\{a_{t,g}\}_{g=1}^{G})$;

Obtain $\{\eta_{t,g}^{\text{base}}\}_{g=1}^{G}$ from f_{base};

for $g = 1$ **to** G **do**

  Set $\eta_{t,g} \leftarrow \eta_{t,g}^{\text{base}} \exp(\alpha_t a_{t,g})$;

end

$\text{OPTIMIZER.STEP}()$;

// The next standard training iteration yields L_{t+1}, ∇_{t+1}; no additional forward/backward is run

Construct next state s_{t+1} from L_{t+1} and ∇_{t+1};

Compute controlled-group-wise rewards $\{r_{t+1,g}\}_{g=1}^{G}$ using L_t, L_{t+1}, and ∇_{t+1};

Compute ψ_{t+1} using Eq. (6) and set $r_{t+1,g}^{\text{final}} \leftarrow r_{t+1,g} - \lambda_{\text{cb}}\psi_{t+1}$ for every group g;

if rank $== 0$ **then**

  Store $(s_t, u_t, a_t, \ell_t^{\text{old}}, r_{t+1}^{\text{final}}, s_{t+1})$, retaining the group axis, in the PPO buffer;

  if *update interval reached* ***or*** *Circuit-Breaker triggered* **then**

    $\pi_\theta.\text{UPDATE}()$;

  end

end

if *Circuit-Breaker triggered* ($\psi_{t+1} = 1$) **then**

  // Abort Training

  Trigger abort signal and broadcast to all workers;

end

B Experimental Details

B.1 Dense Model Setup

We follow the standardized experimental setup introduced by Zhao et al. (2024), with our own scheduler implementations integrated into the same training framework.

For the Llama 2 (Touvron et al., 2023b) models, we use C4 (Raffel et al., 2020) as the pretraining corpus. We fix the batch size to 512 and the maximum sequence length to 256 for all Llama 2 experiments.

We train the following model scales:

- 60M: 11K training steps, corresponding to 1.4B tokens;
- 130M: 20K training steps, corresponding to 2.6B tokens;
- 350M: 60K training steps, corresponding to 7.8B tokens;
- 1B: 100K training steps, corresponding to 13.1B tokens.

Table 4 summarizes the architectural configurations of all Llama 2 dense models used in our experiments.

Table 4: Architecture of Llama 2 dense models.

	60M	**130M**	**350M**	**1B**
d_{model}	512	768	1024	2048
n_{layers}	8	12	24	24
n_{heads}	8	12	16	32
d_{ff}	1376	2048	2736	5461
Vocab	32000	32000	32000	32000

For hyperparameter tuning, we sweep the peak learning rate over the following search grid:

$$\{10^{-4},\, 3\times10^{-4},\, 5\times10^{-4},\, 10^{-3},\, 3\times10^{-3},\, 5\times10^{-3},\, 10^{-2}\}.$$

Alongside the learning rate, we also perform a grid search for the weight decay parameter over:

$$\{10^{-2},\, 5\times10^{-2},\, 10^{-1},\, 2\times10^{-1}\}.$$

B.2 Hyperparameter Selection and Final-PPL Reporting

We use seed 42 to select scheduler configurations and keep the selected values fixed thereafter. Table 1 reports validation PPL at the final update of the seed-52 runs. We evaluate validation PPL every 1K updates, but earlier checkpoints are not used for reporting. Training ends after 11K, 20K, 60K, and 100K updates for the 60M, 130M, 350M, and 1B models. Table 9 lists the repeated runs separately, including the seed-42 tuning runs.

All dense methods share sequence length 256, global batch size 512, gradient clipping at 1.0, and bf16 arithmetic. AdamW uses $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$; Muon uses momentum 0.95, with $\beta = (0.9, 0.95)$ for its AdamW-handled parameters. The common base search covers peak LR $\{10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 10^{-3}, 3\times10^{-3}, 5\times10^{-3}, 10^{-2}\}$ and weight decay $\{0.01, 0.05, 0.1, 0.2\}$. Table 5 gives the selected base settings.

Every configuration in the two-stage screen runs to $0.4T$, after which the top $k = 2$ configurations continue to T. In a completed 28-point calibration study at 60M, the interim and final rankings have Spearman correlations of $\rho = 0.96$ for AdamW and 0.93 for Muon. Advancing two configurations retains the final winner in both cases; advancing only the top interim configuration would increase the selected final PPL by 0.14.

The SOLAR search runs once at 60M and is reused under both optimizers and all target scales. The later 130M sweep evaluates sensitivity around the selected α. The one-time 24-configuration

Table 5: Selected peak LR for each dense-model scale. All settings use weight decay 0.01 and 10% warmup. SOLAR inherits the corresponding Cosine base and does not run a separate base search.

Optimizer	60M	130M	350M	1B
AdamW	3×10^{-3}	1×10^{-3}	1×10^{-3}	5×10^{-4}
Muon	5×10^{-3}	3×10^{-3}	3×10^{-3}	1×10^{-3}

Table 6: Scheduler-specific search at 130M/C4/AdamW. Each method uses the common LR grid unless its update rule determines the LR. No non-scheduler training hyperparameter is retuned.

Method	Selected base LR / WD	Method-specific search → selected
Cosine	$10^{-3}/0.01$	decay to $0.1\times$ peak
WSD	$10^{-3}/0.01$	stable fraction $\{70, 80, 90\}\% \rightarrow 80\%$; decay starts at $0.9T$
CLR	max 10^{-3}, base 10^{-4} / 0.01	period $\{1, 2, 4\}$K → 2K; triangular2
Blockwise LR	$10^{-3}/0.01$	segments $\{2, 3, 4\} \rightarrow 3$; decay $\{0.5, 0.3\} \rightarrow 0.5$
Schedule-Free	$10^{-3}/0.01$	$\beta \in \{0.9, 0.95, 0.98\} \rightarrow 0.9$
Prodigy	adaptive ($d_0 = 10^{-6}$) / 0.01	$d_{\text{coef}} \in \{0.1, 0.5, 1.0\} \rightarrow 1.0$
MECHANIC	$10^{-3}/0.01$	number of scales $\{4, 6\} \rightarrow 6$
AutoLRS	init. $10^{-3}/0.01$	$K = 10$; $\tau_{\text{init}} = 1000$; $\tau_{0,\text{init}} = 100$; $\tau_{\max} = 8000$
AvgLR Replay	inherited / 0.01	no search; replays SOLAR's mean-LR trajectory
SOLAR	inherits Cosine	60M-only 24-configuration search: $\alpha \in \{0.8, 1.0, 1.3, 1.8\} \rightarrow 1.3$; two action-warmup ratios $\rightarrow 0.1T$; initial $\log\sigma \in \{-0.7, 0, 0.7\} \rightarrow 0$. Policy LR 3×10^{-4} and PPO clip 0.2 are fixed.

search costs 7.46 wall-clock hours (59.7 GPU-hours) on 8×RTX 4090. Five 60M meta-training trajectories cost 1.62 wall-hours (13.0 GPU-hours) for AdamW and 1.99 wall-hours (15.9 GPU-hours) for Muon. Across both optimizers, the pre-submission base searches total 616.91 wall-hours and 4,935.3 GPU-hours. AutoLRS repeats candidate forward/backward segments and costs 1.98–2.00× its matched baseline; at 1B it uses 113.47 wall-hours versus 56.75 for Cosine.

B.3 Data, Seeds, Groups, and Recovery State

The C4 validation stream contains 10M held-out tokens and uses `T5TokenizerFast` (`t5-base`, 32K vocabulary). The Pile validation stream contains 1M held-out tokens and uses `LlamaTokenizerFast`. Loss masks padding tokens. Training seeds are $\{42, 52, 62\}$ where repeats are available, and the data-shuffle seed is 42. Each trainable tensor forms one controlled group in the dense implementation, giving $G = 9L + 3 = 75/111/219/219$ groups from 60M to 1B.

SOLAR maintains two rolling checkpoints at a 1K-update cadence. A rollback restores model weights, optimizer state, LR-scheduler state, mixed-precision state, per-rank random-number-generator state, and data-loader position. The PPO state is not rolled back, so the forced update triggered by the failure is retained. Stale rollouts are discarded, followed by a 1K-update cooldown. A forced $8\times$ LR spike reproduces the reference trajectory after restoration to within 3×10^{-7} and yields identical sample hashes. Checkpoint maintenance adds 0.59% wall-clock time at 1B and is included in the full-run accounting in Appendix C.14.

B.4 MoE Model Setup

Qwen2-MoE (Yang et al., 2024) is a strong open-source Mixture-of-Experts (MoE) decoder-only Transformer. Compared with the dense Llama 2 models used in our main experiments, its sparse routing mechanism introduces a more challenging optimization landscape, making it a useful testbed for evaluating online learning-rate scheduling under highly non-stationary training dynamics. For our MoE experiments, we train on **The Pile** (Gao et al., 2020) and build on the Qwen2-MoE codebase released by LITE (Zhu et al., 2026). To align the MoE setting with the Llama 2 experiments, we disable sliding-window attention and use a maximum sequence length of 256 with batch size 512.

Table 7: Architecture of the Qwen2-MoE 1B model.

Parameter	d_{model}	n_{layers}	n_{heads}	$n_{\text{kv_heads}}$	d_{ff}	$d_{\text{moe_ff}}$	$d_{\text{shared_ff}}$	n_{experts}
Value	768	15	12	12	3072	768	3072	32

The model is configured to activate 4 experts per token via a top-k routing strategy ($k = 4$), with 32 experts in total. To encourage stable expert utilization and mitigate representation collapse, we apply a load-balancing auxiliary loss with coefficient 0.01 and a router z-loss with coefficient 0.001. We train for 100K steps (approximately 13.1B tokens). Table 7 summarizes the full architectural configuration.

B.5 Optimizer Configurations

Methods that replace or modify optimizer behavior are described here, whereas external LR schedules are described in Appendix B.6.

AdamW. All Adam-family experiments use the default PyTorch AdamW (Loshchilov & Hutter, 2019) implementation with $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$. The peak learning rate and weight decay are tuned via grid search as described in Appendix B.1.

Muon. Our Muon implementation follows the standard configuration described by Liu et al. (2025). It uses momentum 0.95, Nesterov acceleration, and 5 Newton-Schulz orthogonalization steps. The learning rate for matrix-shaped parameters is scaled by $0.2\sqrt{\max(A, B)}$, where (A, B) are the parameter dimensions. Scalar and embedding parameters are handled by an internal AdamW with $\beta = (0.9, 0.95)$ and $\epsilon = 10^{-8}$.

Prodigy. We use the official Prodigy (Mishchenko & Defazio, 2024) implementation without an external learning-rate scheduler. We select d_{coef} using the grid in Table 6 with weight decay fixed at 0.01; the remaining hyperparameters use their defaults. Because the released implementation is AdamW-based, we report Prodigy only under the AdamW group and omit it from Muon.

Schedule-Free AdamW. Schedule-Free AdamW (Defazio et al., 2024) is evaluated using the official implementation. We tune the base learning rate on the common validation grid and search $\beta \in \{0.9, 0.95, 0.98\}$, selecting $\beta = 0.9$; weight decay is fixed at 0.01 and the remaining settings follow the official recipe. Because the released implementation is AdamW-based, we report Schedule-Free only under the AdamW group and omit it from Muon.

B.6 Scheduler Baselines

We compare SOLAR against the scheduler baselines used in the main paper. All baselines are implemented in the same training codebase and evaluated under the same data, model, batch size, sequence length, and optimizer. Unless otherwise specified, all warmup-decay schedules use the same minimum LR ratio 0.1. For methods with additional hyperparameters, we tune them using the same validation setup.

Cosine schedule. Warmup-Cosine-Decay (Cosine) (Loshchilov & Hutter, 2017) uses a linear warmup during the first 10% of training and then applies cosine annealing to decay the learning rate from $\eta_{\max}$ to $0.1\eta_{\max}$ over the remaining steps. We tune the peak learning rate and weight decay within the same validation setting as the other baselines.

AvgLR Replay baseline. This diagnostic baseline is placed immediately after Cosine because it replays SOLAR's scalar average trajectory on the same base scheduling interface, serving as a mechanism test rather than a conventional strong baseline. To construct it, we first record the realized average learning-rate trajectory of a SOLAR run. At each step t, we compute the unweighted mean

Algorithm 3: Warmup-Cosine-Decay baseline

Input: Base optimizer, peak LR $\eta_{\max}$, total steps T, warmup ratio $\rho_w = 0.1$

$T_w \leftarrow \lfloor \rho_w T \rfloor$, $T_c \leftarrow T - T_w$;
for *update step* $t = 0, 1, \ldots, T-1$ **do**
  if $t < T_w$ **then**
    $\eta_t \leftarrow \eta_{\max} \cdot \frac{t+1}{T_w}$;
  else
    $u \leftarrow t - T_w$;
    $\eta_t \leftarrow 0.1\eta_{\max} + \frac{1}{2}(\eta_{\max} - 0.1\eta_{\max})\left[1 + \cos\left(\pi \frac{u}{T_c}\right)\right]$;
  Set the optimizer learning rate to η_t and run one update;

Algorithm 4: Warmup-Stable-Decay baseline

Input: Base optimizer, peak LR $\eta_{\max}$, total steps T

$T_w \leftarrow \lfloor 0.1T \rfloor$, $T_s \leftarrow \lfloor 0.8T \rfloor$, $T_d \leftarrow T - T_w - T_s$;
for *update step* $t = 0, 1, \ldots, T-1$ **do**
  if $t < T_w$ **then**
    $\eta_t \leftarrow \eta_{\max} \cdot \frac{t+1}{T_w}$;
  else if $t < T_w + T_s$ **then**
    $\eta_t \leftarrow \eta_{\max}$;
  else
    $u \leftarrow t - (T_w + T_s)$;
    $\eta_t \leftarrow \eta_{\max} - (\eta_{\max} - 0.1\eta_{\max}) \frac{u+1}{T_d}$;
  Set the optimizer learning rate to η_t and run one update;

across controlled parameter groups:

$$\bar{\eta}_t = \frac{1}{G} \sum_{g=1}^{G} \eta_{t,g}^{\mathrm{SOLAR}}.$$

We then replay this scalar trajectory step by step during baseline evaluation and assign the same learning rate to every controlled parameter group:

$$\eta_{t,g} = \bar{\eta}_t, \quad \forall g.$$

This baseline uses the SOLAR trajectory directly and does not apply any group-wise weighting.

WSD schedule. Warmup-Stable-Decay (WSD) (Hu et al., 2024) uses linear warmup during the first 10% of training, a stable plateau for the next 80%, and a final linear decay during the last 10%. During the decay phase, the learning rate decreases from $\eta_{\max}$ to $0.1\eta_{\max}$.

CLR schedule. CLR (Smith, 2017) cyclically varies the learning rate between a lower and an upper bound. We implement CLR as a schedule that can be plugged into the base optimizer used in each experiment, so the training loop uses the same `optimizer.step()` interface as the other baselines. We use a linear warmup from $\eta_{\min}$ to $\eta_{\max}$, followed by a cyclic schedule. Unless otherwise stated, we use $\eta_{\min} = 0.1\eta_{\max}$, the `triangular2` policy, and a half-cycle length of 1000 optimizer update steps. For hyperparameter tuning, we search the peak learning rate, weight decay, and scheduler-specific settings under the same validation setting as the other baselines.

Blockwise LR. Blockwise LR (Wang et al., 2025) assigns block-type-specific learning rates based on the transformer sharpness disparity principle. We use the official AdamW-based implementation, tune the peak learning rate on the common validation grid, and search the number of segments over $\{2, 3, 4\}$ and the decay factor over $\{0.5, 0.3\}$, selecting three segments and a decay factor of 0.5; weight decay is fixed at 0.01. Because the released implementation is AdamW-based, we report Blockwise LR only under the AdamW group and omit it from Muon.

Algorithm 5: CLR

Input: Base optimizer, peak LR $\eta_{\max}$, lower ratio r, warmup steps T_w, half-cycle length s

$\eta_{\min} \leftarrow r\eta_{\max}$;
for *update step* $t = 0, \ldots, T-1$ **do**
  if $t < T_w$ **then**
    $\eta_t \leftarrow \eta_{\min} + (\eta_{\max} - \eta_{\min})\frac{t+1}{T_w}$;
  else
    $u \leftarrow t - T_w,\ c \leftarrow \lfloor \frac{u}{2s} \rfloor$;
    $x \leftarrow \left| \frac{u}{s} - 2c - 1 \right|, a \leftarrow \max(0, 1-x)$;
    $a \leftarrow a/2^c$;
    $\eta_t \leftarrow \eta_{\min} + (\eta_{\max} - \eta_{\min})a$;
  Set the optimizer learning rate to η_t and run one update;

Algorithm 6: AutoLRS

Input: Base optimizer, LR interval $[\eta_{\min}, \eta_{\max}]$, number of BO trials K, stage length τ, probe length τ_0

while *training is not finished* **do**
  Buffer the next τ update batches and save model, optimizer, and RNG states;
  Reset the Bayesian optimizer for the current stage;
  for $i = 1, \ldots, K$ **do**
    Suggest candidate η_i by LCB acquisition with exploration coefficient κ over $[\log \eta_{\min}, \log \eta_{\max}]$;
    Restore the saved state;
    Train for τ_0 buffered update steps with global LR η_i;
    Record the short-horizon loss curve;
    Fit the exponential forecaster and predict the stage loss $\hat{\ell}_i$;
    Update the GP with $(\log \eta_i, \hat{\ell}_i)$;
  Select $i^\star = \arg\min_i \hat{\ell}_i$;
  Restore the saved state and train the real model for τ steps with global LR $\eta_{i^\star}$;
  Increase τ up to $\tau_{\max}$ and set $\tau_0 \leftarrow \max(1, \tau/10)$;

AutoLRS schedule. AutoLRS (Jin et al., 2021) selects a piecewise-constant learning rate online through repeated short-horizon candidate evaluation. At the beginning of each stage, AutoLRS buffers the next τ update batches, checkpoints the model, optimizer, and random-number-generator states, and evaluates K candidate learning rates. Each candidate is trained for only τ_0 update steps from the same checkpoint and on the same buffered data. The short-horizon loss curve is then summarized by an exponential forecaster and used as the objective for Bayesian optimization. After the K trials, AutoLRS restores the original checkpoint and trains the real model for the full τ-step stage using the candidate LR with the lowest predicted loss.

We use $K = 10$, $\kappa = 1000$ (the LCB exploration coefficient), $\tau_{\text{init}} = 1000$, $\tau_{0,\text{init}} = 100$, and $\tau_{\max} = 8000$. The stage length doubles after each stage until it reaches $\tau_{\max}$, and τ_0 is set to one tenth of the current stage length. During early stages, the objective is the training loss curve; once the stage length reaches $\tau_{\max}$, we switch to a fixed validation subset of 4 pre-cached C4 validation batches. AutoLRS assigns the selected LR to all parameter groups.

MECHANIC schedule. MECHANIC (Cutkosky et al., 2023) learns a single model-wide scale that rescales the cumulative update direction produced by the base optimizer. The method maintains a reference copy of the initial parameters, a cumulative base-update direction, and a collection of tuner states that determine the next global scale. At each step, MECHANIC first applies one update of the base optimizer, then computes a scalar feedback signal from the current gradient, the old cumulative update direction, and global norms. This feedback is used to update the tuner states and obtain a new

scale S_{t+1}. Finally, the parameters are reparameterized around the reference point as

$$x_{t+1} = x_{\text{ref}} + S_{t+1}\Delta_{t+1}.$$

We use the official MECHANIC implementation under the same training setup as the corresponding baseline. The base learning rate and weight decay follow the common validation search, and Table 6 reports the number-of-tuners search. The remaining MECHANIC settings follow the official implementation.

B.7 Hardware

All dense and Qwen2-MoE experiments up to 1B parameters run on $8\times$NVIDIA RTX 4090 GPUs. The DeepSeek-V2-style 3B MoE experiment runs on $32\times$NVIDIA A800 80GB GPUs. All dense and 1B MoE wall-clock and overhead measurements use the RTX 4090 system; the 3B MoE timing uses the A800 system.

C Additional Ablation and Experimental Results

Unless otherwise specified, the following ablations use the seed-52 run of a 130M-parameter Llama 2 model trained on C4 with AdamW under Appendix B.1. They isolate the state, action, reward, control granularity, safety, and transfer choices used by SOLAR.

C.1 Positioning among Learned LR Controllers

Table 8 distinguishes the object learned by each controller and the runs used to acquire it. Prior work establishes RL-based global scaling, state-conditioned actions, transfer, and layer-wise control. SOLAR targets a different acquisition regime: the controller learns during the same full autoregressive LLM pretraining trajectory that it must improve.

Table 8: Positioning among representative learned LR controllers. "Same live run" means that policy updates occur inside the final run being improved.

Method	Learned object	Acquisition	Evaluated setting
Daniel et al. (2016)	global step size	repeated episodes	small networks
Xu et al. (2017)	global absolute LR	actor–critic with resets	vision
Xu et al. (2019)	global LR profile	past complete histories	Fashion-MNIST, CIFAR-10
GNS (2022)	global graph policy	separate target episodes	vision, GLUE
Subramanian et al. (2023)	PPO LR schedule	separate controller training	MNIST, CIFAR-100
GANNO (2023)	layer-wise absolute LR	separate environments	vision
SOLAR-online	bounded group residual	same live run	autoregressive LLM pretraining
SOLAR-frozen	bounded group residual	source proxy runs	dense scale/corpus transfer

C.2 Repeated Final-Checkpoint Results

Table 9 lists final PPL for every repeated run and reports the mean across available seeds. Hyperparameters remain fixed after selection. The two SOLAR-frozen rows reuse one source-trained policy across all three target seeds.

Table 9: Final-checkpoint PPL by seed. Seed 42 is the selection seed for online and baseline configurations; seeds 52 and 62 use the selected configuration. SOLAR-frozen reuses one source-trained policy across all target seeds.

Setting	Method	Seed 42	Seed 52	Seed 62	Mean PPL
60M AdamW	Cosine	30.71	30.49	30.34	30.51
	WSD	29.63	29.80	29.91	29.78
	SOLAR-online	28.99	28.92	29.14	**29.02**
130M AdamW	Cosine	24.41	24.52	24.57	24.50
	WSD	24.07	23.96	23.90	23.98
	SOLAR-online	22.95	22.79	22.84	**22.86**
	SOLAR-frozen	22.96	22.41	22.90	**22.76**
130M Muon	Cosine	22.61	22.55	22.44	22.53
	SOLAR-online	22.11	21.99	22.06	**22.05**
	SOLAR-frozen	22.24	21.87	22.16	**22.09**
350M AdamW	Cosine	18.42	18.31	18.47	18.40
	SOLAR-online	17.19	17.23	17.12	**17.18**
1B AdamW	Cosine	16.44	16.52	–	16.48
	SOLAR-online	14.90	14.83	–	**14.87**

SOLAR is lower in every paired 350M and 1B run. The paired gaps are 1.23/1.08/1.35 PPL at 350M and 1.54/1.69 at 1B. At 130M, online/frozen mean PPLs are 22.86/22.76 under AdamW and 22.05/22.09 under Muon.

C.3 Online Acquisition and Frozen Reuse

SOLAR supports in-run acquisition and amortized reuse of the same controller design. SOLAR-online updates the policy inside the target run. SOLAR-frozen reuses a policy acquired over K full-length 60M online runs and performs no target PPO updates. Freezing fixes all source-policy parameters, including the final learned shared $\log \sigma$, not the action sequence: the controller continues to recompute group-wise actions from each target state.

Table 10: Source-side acquisition and 130M/C4/AdamW frozen reuse. Every target evaluation uses seed 52. The $K = 0$ row keeps the policy at its random initialization.

Full-length 60M source runs K	Frozen final PPL
0	25.04
1	23.26
2	22.89
3	22.58
5	**22.41**

Frozen reuse improves as the policy is acquired over more source runs, moving from 25.04 PPL at $K = 0$ to 22.41 at $K = 5$. Each source run is a complete online pretraining trajectory.

Table 11: Online acquisition and frozen reuse across source–target settings. Rows with C4 targets use seed 52; the Pile-target row reports the mean of seeds 42 and 52.

Source → target	Cosine	SOLAR-frozen	SOLAR-online
60M/C4/AdamW → 130M/C4/AdamW	24.52	**22.41**	**22.79**
60M/C4/Muon → 130M/C4/Muon	22.55	**21.87**	**21.99**
60M/C4/AdamW → 130M/Pile/AdamW	13.96	**13.21**	**12.99**
60M/C4/AdamW → 130M/C4/Muon	22.55	**22.49**	**21.99**

Both SOLAR modes remain below Cosine in the optimizer-matched rows. Frozen reuse reaches the lower PPL on the matched C4 targets, while online acquisition reaches the lower PPL under the corpus and optimizer changes.

C.4 Matched Learned-Controller Comparisons

The 130M/C4/AdamW controls in Table 12 share 20K updates, weight decay 0.01, and a tuned peak LR of 10^{-3} where a base schedule is used. N1–N3 are design-axis probes; N4 is an independent LLM adaptation of GANNO with its own controller grid. Each control is tuned on its own search space.

Implementations and tuning protocol. N1 and N2 use the same global, state-conditioned PPO controller with $G = 1$. N1 applies a bounded multiplier that re-anchors to the shared base schedule at every update; N2 compounds each multiplier on the preceding LR. N3 retains the anchored group-wise interface but replaces PPO with a myopic Baydin-style hypergradient rule. These three controls are trained directly in the 130M target run, and each is searched on its own controller grid. N4 follows the GANNO control structure in a separate implementation: a shared recurrent IPPO actor–critic serves 15 module-level agents, chooses among nine categorical actions on cumulative absolute LRs around $\eta^* = 10^{-3}$ every 50 updates, and uses counterfactual difference rewards. Its policy is meta-trained at 60M, frozen, and evaluated at 130M with no target PPO updates; its controller is searched on its own grid. N1–N3 isolate individual design axes, whereas N4 is a complete competing controller adapted to the LLM setting.

Table 12: Matched controls that isolate the SOLAR control structure. The matched SOLAR/N1/N2 comparisons use seeds 42 and 52; each arrow gives the mean. Three-run static-baseline summaries include the sample standard deviation.

Method and control structure	Per-seed PPL → summary	Isolated axis
SOLAR-online: anchored group residual, $G = 111$	$22.95/22.79 \rightarrow \mathbf{22.87}$	full method
N1 Global-PPO-Residual: anchored, $G = 1$	$23.68/23.80 \rightarrow 23.74$	group granularity
WSD: tuned static base	$24.07/23.96/23.90 \rightarrow 23.98 \pm 0.09$	learned control
N3 Group-MHD: anchored, group-wise, non-RL	$24.31/24.21 \rightarrow 24.26$	learned feedback
Cosine: tuned static base	$24.41/24.52/24.57 \rightarrow 24.50 \pm 0.08$	learned control
N4 GANNO-IPPO: layer-wise absolute LR	$24.89/25.01 \rightarrow 24.95$	residual design
N2 Global-PPO-Recursive: unanchored, $G = 1$	$27.30/26.88 \rightarrow 27.09$	anchoring and bounds

N1 and N2 use the same global, state-conditioned PPO interface. On the common seeds 42 and 52, re-anchoring each multiplier to the base and bounding the action improves 27.09 to 23.74, a 3.35-PPL change. Moving from one global action to $G = 111$ group actions further improves PPL to 22.87; the third SOLAR run gives a three-run mean of 22.86 ± 0.08 in Table 9. N3 shows that a group-wise residual without a learned state-conditioned loop does not recover the same gain. N4 provides a layer-wise RL controller adapted to the same LLM setting. Under 60M-to-130M frozen-policy transfer, N4 reaches 24.95 PPL and SOLAR-frozen reaches a three-run mean of 22.76, a 2.19-PPL gap between the two LLM controller adaptations. Within SOLAR, replacing the residual action with a target-trained absolute-LR action changes 22.79 to 24.74, a 1.95-PPL gap. With the SOLAR architecture held fixed, untrained, near-deterministic, and state-blind policies reach 25.04, 25.03, and 25.21; AvgLR Replay reaches 25.61 at the final checkpoint. Together, these controls isolate the contribution of the anchored residual interface, state-conditioned learning, and group-wise actions.

Failure-mode diagnostics. The N1/N2 pairing exposes a slow failure that a loss-spike safeguard does not detect. In an $\alpha = 2.6$ stress run ($n = 1$), N2 produces no loss spike: its recursively updated

LR drifts to a $5.7\times$ mean multiplier and finishes at 79.58 PPL. N3 exposes a second failure mode. Moving its single hypergradient coefficient by one decade changes PPL at the $0.4T$ screen from 31.05 to 47.31, showing sharp sensitivity to its local update scale.

Action-bound sensitivity. The main setting $\alpha = 1.3$ is inherited from the 60M search; Table 13 evaluates sensitivity around this setting at 130M. Performance remains better than Cosine over a twofold range, and the Circuit-Breaker first activates at $\alpha = 1.8$.

Table 13: Sensitivity to the residual action bound on 130M/C4/AdamW.

α	0.8	1.0	1.3	1.6	1.8
Final PPL	23.42	23.05	**22.79**	22.94	23.61
CB activation	no	no	no	no	yes

C.5 Frozen-Policy Robustness and μP Compatibility

Target base-LR sweep. We freeze a 60M/C4 policy and apply it with no target PPO updates. Table 14 varies the target peak LR over a fourfold window. Each cell is a single seed-52 run; the experiment measures sensitivity to the supplied base LR rather than run-to-run variance.

Table 14: Frozen residual-policy transfer under misspecified target base LRs.

Base peak	130M Cosine	130M frozen	350M Cosine	350M frozen
$0.5\times$	25.05	**22.72**	18.79	**17.61**
$1.0\times$	24.52	**22.41**	18.31	**17.37**
$2.0\times$	diverged @4.1K	**22.68**	diverged @6.8K	**17.96**

The frozen policy beats the same-base Cosine run in all six settings and completes both $2\times$ runs. At 130M, the negative-action fraction changes from 0.29 to 0.47 to 0.73 as the base rises from $0.5\times$ to $2\times$, showing bidirectional state-conditioned correction rather than a fixed positive multiplier.

The main-table protocol searches LR/WD at each target scale, at costs of 14.02, 35.66, and 227.65 wall-clock hours for 130M, 350M, and 1B. Freezing reuses the resulting base schedule and removes target PPO training and target-scale search over SOLAR-specific hyperparameters. The avoided controller-side work is $0.9\times$, $2.7\times$, and $2.7\times$ the corresponding LR/WD-search cost.

μP width transfer. We combine SOLAR with maximal-update parameterization (μP) (Yang et al., 2021). A 71M base and 130M target differ only in hidden width; both use 12 layers, head dimension 64, $G = 111$, sequence length 256, global batch 512, 20K updates, weight decay 0.01, and 10% warmup. A three-point search at 71M selects $\eta^* = 2 \times 10^{-3}$. μP transfers per-tensor target LRs from fan-in: 2.000×10^{-3} for embeddings/RMSNorm and approximately 1.333×10^{-3} for attention, MLP, and readout tensors. No LR/WD search occurs at 130M.

Table 15: SOLAR on a directly searched standard-parameterization (SP) base and a μP-transferred base at 130M/C4/AdamW. For the two-run μP settings, values before the arrow are per-seed final PPL and the arrow gives their mean.

Method	Searched SP base	μP-transferred base
Cosine	24.50 ± 0.08 ($n = 3$)	$24.44/24.53 \rightarrow 24.49$
SOLAR-online	22.86 ± 0.08 ($n = 3$)	$22.87/22.94 \rightarrow \mathbf{22.91}$
SOLAR-frozen	22.41 ($n = 1$)	$22.62/22.74 \rightarrow \mathbf{22.68}$

μP+Cosine reaches 24.49 PPL, compared with 24.50 for the searched SP base. SOLAR improves each paired μP run by 1.57–1.59 PPL online and 1.79–1.82 frozen. The frozen configuration performs

neither LR/WD search nor PPO updates at the 130M target. It inherits $\alpha = 1.3$, action warmup $0.1T$ (= 2K updates), the source policy's final learned $\log \sigma$, and all reward coefficients. The online policy's negative-action fraction is 0.44 under μP and 0.47 under SP.

We validate the μP implementation by checking activation-scale consistency across widths. Across hidden widths 256–1024, the mean absolute activation at each layer changes by less than 6% under μP; under standard parameterization, the width-1024 value is approximately $2.1\times$ the width-256 value.

Corpus shift. We also train 130M models on The Pile with the same architecture and token budget. The Cosine peak LR is re-searched on The Pile over $\{5 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$; 10^{-3} is selected and 3×10^{-3} diverges at 3.4K updates. Table 16 reports the two final PPL values for each method.

Table 16: Cross-corpus evaluation on 130M/The Pile/AdamW.

Method	Seed 52	Seed 42	Mean	Gain vs. Cosine
Cosine	13.90	14.02	13.96	–
SOLAR-online	12.94	13.03	**12.99**	0.97 (6.9%)
SOLAR-frozen (60M/C4 policy)	13.18	13.24	**13.21**	0.75 (5.4%)

The online policy preserves its C4 relative gain under the corpus shift. On The Pile, online has a 0.22 lower mean PPL than frozen across seeds 42 and 52; on the seed-52 C4 run, frozen is 0.38 PPL lower than online. Both modes remain below their matched Cosine baselines.

C.6 Static and Open-Loop Replay Controls

We derive replay controls from a completed 130M/C4/AdamW/seed-52 SOLAR-online run. The static group profile averages each of the $G = 111$ multipliers over 20K updates and normalizes their cross-group mean to one. The time-varying replay applies the recorded multiplier for every group and update. The level-corrected scalar replay rescales SOLAR's step-wise mean LR before replay. Each replay runs open-loop without a policy, state input, PPO update, or Circuit-Breaker. A five-point global rescaling search for the static profile selects $c = 1.0$ and retains its 24.19 final PPL.

Table 17: Replay controls on 130M/C4/AdamW with seed 52. All entries report final PPL.

Method	Group-wise	Time-varying	Live state	Final PPL
Cosine	–	base only	–	24.52
AvgLR Replay	–	✓	–	25.61
Level-corrected scalar replay	–	✓	–	24.63
Static normalized group profile	✓	–	–	24.19
SOLAR-online	✓	✓	✓	**22.79**

AvgLR Replay and the level-corrected scalar replay reach 25.61 and 24.63 PPL. Static group allocation improves Cosine from 24.52 to 24.19 PPL, while SOLAR-online reaches 22.79.

Table 18: SOLAR-derived replays across seeds, base LRs, and corpora. Both replay profiles come from the seed-52 C4 run at the selected base LR. The Pile row uses seed 52; "–" denotes a setting not evaluated.

Setting	Cosine	Static group	Time-varying group	SOLAR-frozen	SOLAR-online
C4, seed 42, 1.0× base	24.41	24.29	23.94	22.96	**22.95**
C4, seed 52, 0.5× base	25.05	24.83	24.47	**22.72**	–
C4, seed 52, 2.0× base	div. @4.1K	div. @3.8K	div. @4.4K	**22.68**	–
Pile, seed 52, 1.0× base	13.90	13.71	13.55	13.18	**12.94**

The learned policies remain below both replay controls across the seed, base-LR, and corpus shifts. At the $2.0\times$ base LR, SOLAR-frozen completes training at 22.68 PPL, while Cosine and both open-loop replays diverge.

C.7 Importance of Global and Local State Features

To evaluate our state formulation, we compare the **Full state** (the complete 10-dimensional state described in Appendix A.1) against two variants:

- **Global-only state**: Omits per-group statistics, relying solely on the 4 shared global features (e.g., training step, global loss).
- **Local-only state**: Omits macro-phase indicators, relying solely on the 6 per-group local features (e.g., layer-wise gradient norms).

Combining global and local features gives the lowest final PPL in Table 19: 22.79, compared with 23.11 for global-only and 23.57 for local-only.

Table 19: Ablation on state representation for the 130M model.

State Variant	Final Eval PPL (↓)
Full state (Ours)	22.79
Global-only	23.11
Local-only	23.57

C.8 Stochastic Exploration vs. Near-Deterministic Scheduling

SOLAR relies on stochastic exploration to gather informative credit assignment signals under noisy, delayed loss feedback. To quantify the contribution of stochastic exploration, we fix the policy standard deviation to a near-zero constant σ_{det} throughout training while retaining the same PPO update and residual action parameterization:

$$u_{t,g} = \mu_{t,g} + \sigma_{\text{det}}\epsilon_{t,g}, \quad \epsilon_{t,g} \sim \mathcal{N}(0,1), \qquad \sigma_{\text{det}} \approx 0,$$
$$\widetilde{a}_{t,g} = \tanh(u_{t,g}), \qquad a_{t,g} = \text{clip}(\widetilde{a}_{t,g}, -1 + 10^{-4}, 1 - 10^{-4}).$$

The resulting actions are effectively determined by the policy mean. All other components—the residual LR modulation ((3)), action-scale warmup schedule, PPO surrogate objective, value function, entropy regularization, and network architecture—remain identical to the stochastic policy.

Suppressing stochasticity increases final PPL from 22.79 to 25.03 in Table 20. The result is consistent with stochastic sampling improving exploration under delayed feedback.

Table 20: Comparison of stochastic exploration versus near-deterministic scheduling (130M model).

Policy Formulation	Final Eval PPL (↓)
Stochastic (Ours)	22.79
Near-deterministic	25.03

C.9 Residual LR Modulation versus Direct Absolute LR Prediction

A natural baseline for RL-based learning-rate scheduling is to predict the absolute learning rate directly at each step. We instantiate this idea as a **Direct-Abs** baseline. Specifically, the scheduler first samples an unconstrained latent action,

$$u_{t,g} \sim \mathcal{N}(\mu_{t,g}, \sigma^2), \qquad \widetilde{a}_{t,g} = \tanh(u_{t,g}), \qquad a_{t,g} = \mathrm{clip}(\widetilde{a}_{t,g}, -1 + 10^{-4}, 1 - 10^{-4})$$

where $\mu_{t,g}$ is the scheduler output and the scalar σ is a shared learnable parameter controlling action stochasticity. The squashed action $a_{t,g}$ is then mapped to an absolute learning rate in log-space:

$$\log \eta_{t,g} = \log \eta_{\min} + \frac{a_{t,g} + 1}{2} \left(\log \eta_{\max} - \log \eta_{\min}\right),$$

and the final learning rate is obtained by exponentiation,

$$\eta_{t,g} = \exp(\log \eta_{t,g}).$$

This parameterization keeps the action bounded and well-defined, while making the scheduler operate directly on the absolute learning-rate scale. We set $\eta_{\min} = 0.1\eta_{\max}$ and select $\eta_{\max}$ from the same seven-point LR grid used for the other schedulers.

Despite this tuning, Direct-Abs remains brittle in practice. Because the scheduler is responsible for both the absolute scale and the temporal evolution of the learning rate from the very beginning of training, early miscalibrated actions can induce overly aggressive updates, destabilize optimization, and even trigger pronounced loss spikes. Once such disruptions occur in the early stage, they are often difficult to recover from, which ultimately degrades final performance.

SOLAR therefore adopts residual LR modulation on top of a base scheduler. Specifically, the scheduler predicts a bounded multiplicative adjustment:

$$\eta_{t,g} = \eta_{t,g}^{\mathrm{base}} \cdot \exp(\alpha_t a_{t,g})$$

where $a_{t,g}$ is the scheduler action and α_t controls the residual modulation strength. To further stabilize early training, we linearly warm up the residual scale α_t from 0 to α, which limits the magnitude of residual perturbations before the scheduler becomes reliable.

The combined residual-and-warmup design reaches 22.79 final PPL, compared with 24.74 for Direct-Abs under the matched LR search (Table 21).

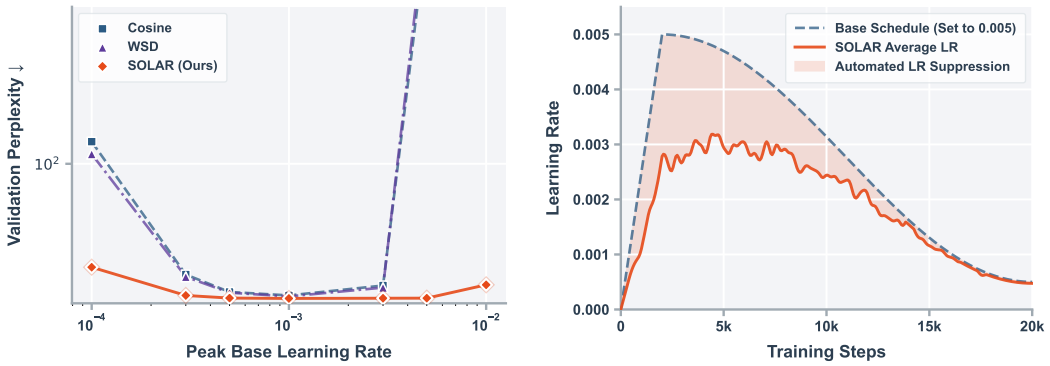
Table 21: Ablation on residual LR modulation versus direct absolute LR prediction (130M model).

Design	Final Eval PPL (↓)
SOLAR (Residual + Warmup)	22.79
Direct-Abs	24.74

C.10 Robustness to Suboptimal Base Learning Rates

Figure 5a complements the matched $0.5\times/1\times/2\times$ transfer study in Appendix C.5 with a wider sensitivity sweep on the 130M AdamW setting. We vary the peak base LR from 10^{-4} to 10^{-2} for Cosine, WSD, and SOLAR with a Cosine base.

Cosine and WSD deteriorate once the peak LR exceeds their selected operating region and diverge at the largest values. SOLAR remains trainable over a wider observed interval and, at high base



(a) Final PPL across peak base LRs on the 130M AdamW setting. Missing high-LR points denote diverged runs.

(b) At a peak base LR of 0.005, negative residual actions reduce the effective LR below the supplied base schedule.

Figure 5: Sensitivity analysis and mechanism illustration for SOLAR.

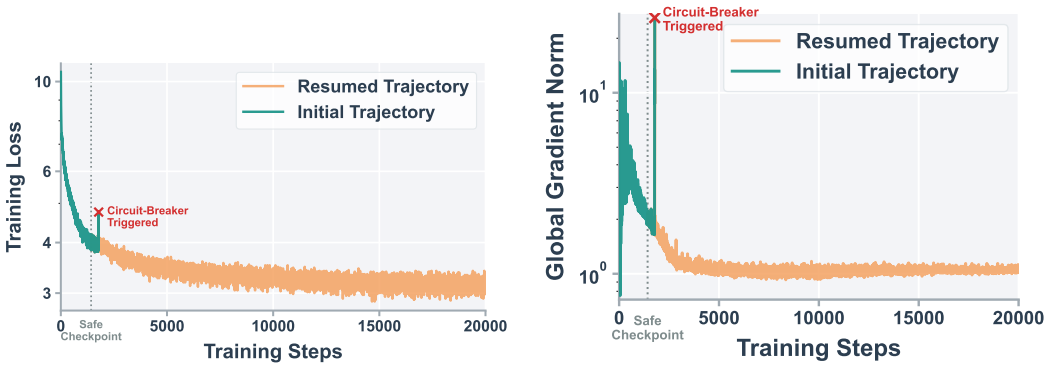
LRs, uses negative residual actions to reduce the effective LR. Across the evaluated sweep, SOLAR tolerates substantial base-LR misspecification and actively compensates when the supplied LR is too large.

Observed response. Figure 5b records the controller response at a peak base LR of 0.005. As gradient statistics worsen, the policy outputs $a_{t,g} < 0$ for affected groups. The mapping $\eta_{t,g} = \eta_{t,g}^{\text{base}} \exp(\alpha_t a_{t,g})$ then lowers their effective LRs without replacing the long-horizon base profile. This trace provides a direct mechanism for the wider stable interval observed in Figure 5a.

C.11 Efficacy of the Automated Circuit-Breaker Mechanism

The Circuit-Breaker handles rare optimization anomalies by restoring a complete training state. None of the main-table or MoE runs triggers it. We therefore validate recovery separately with a stress test.

Experimental Setup. We stress-test the Circuit-Breaker on the 130M model by increasing the action bound to $\alpha = 1.8$, which makes large LR multipliers more likely.



(a) Training loss trajectories.

(b) Global gradient norm trajectories.

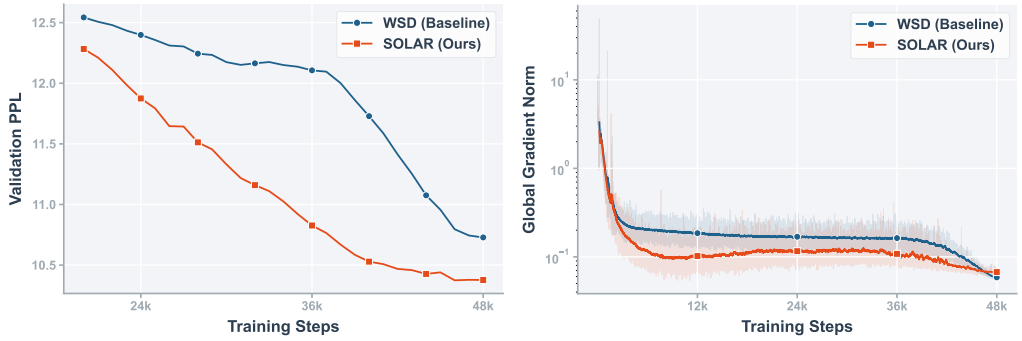
Figure 6: Demonstration of the automated Circuit-Breaker mechanism under an aggressive exploration stress test ($\alpha = 1.8$). In both panels, the teal line represents the initial training rollout, which experiences a severe spike and triggers the Circuit-Breaker (marked by the red cross). The orange line represents the automatically resumed trajectory, which restarts from the last safe checkpoint (indicated by the vertical dotted line) and successfully completes the training process.

Results and Analysis. As illustrated in Figure 6, the initial training rollout (teal line) proceeds normally until the agent samples an overly aggressive learning rate multiplier, causing a sudden and severe spike in both the training loss and the global gradient norm.

In a conventional pretraining pipeline, such an event would terminate the run, requiring manual inspection, learning-rate reduction, and a restart from a checkpoint. SOLAR handles this autonomously: the Circuit-Breaker detects the anomaly ($\kappa_{t+1} > 1.5$), aborts the current step, and applies the global Circuit-Breaker penalty to the RL agent. The scheduler then performs a forced PPO update using this failure experience before resuming the model and optimizer from the last safe checkpoint while retaining the updated PPO state.

After the forced PPO update and rollback, the resumed trajectory passes the failure point and completes training without another severe spike. This stress test verifies the automated recovery path under aggressive exploration.

C.12 Additional Validation on the DeepSeek-V2 3B MoE Model in Megatron



(a) Validation PPL (b) Global grad norm

Figure 7: Additional validation on the DeepSeek-V2 3B MoE model in Megatron. SOLAR is compared against a matched WSD baseline under the same model, data, tokenizer, optimizer, and training setup.

We run a separate 3B-scale MoE experiment following the DeepSeek-V2 (DeepSeek-AI et al., 2024) architecture in Megatron (Shoeybi et al., 2019). This setting changes the model family, corpus, sequence length, distributed stack, and hardware relative to the main dense study.

The model uses sandwich normalization, multi-latent attention, 12 Transformer layers, hidden size 1280, 64 experts, and top-6 routing. The first layer is dense, and the remaining 11 layers use MoE blocks. We train in bf16 with sequence length 4096, global batch size 1024, and 48K updates, or approximately 201B token positions. The corpus is Dolma 3 Mix (Team OLMo et al., 2025). Both methods use the same corpus, tokenizer, model configuration, optimizer, token budget, and 32×A800 80GB hardware. The WSD baseline uses 2K warmup updates, 12K decay updates, a grid-searched peak LR of 8.6×10^{-4}, and a minimum LR of 7×10^{-6}. SOLAR uses this same WSD schedule as its base and updates only the residual controller online.

Figure 7 reports validation PPL and global gradient norm. SOLAR reaches a final PPL of 10.38, compared with 10.73 for WSD, and the observed gap peaks near 1.33 around update 38K. Its gradient-norm trace is also lower and smoother over most of the run. We run each method once in this setting. The comparison tests portability to a different MoE stack but does not estimate run-to-run variance.

C.13 Control Experiments

These controls complement the replay analyses in Section 5.3 and Appendix C.6. They compare SOLAR with state-independent stochastic actions and an untrained fixed-policy controller under the same 130M Llama 2 AdamW setup, base schedule, action-scale warmup, and training budget.

C.13.1 State-Agnostic Stochastic Scheduler

To evaluate state-conditioned policy learning under the same stochastic residual family, we keep the multiplicative residual mapping but replace the learned state-conditioned action with $a_{t,g} \sim \mathrm{TanhNormal}(0, \sigma)$. We select σ from $\{1, 2, 3, 4, 5\}$ on the tuning seed and report that configuration's final checkpoint. The base schedule, action-scale warmup, and training setup match SOLAR.

The selected state-agnostic scheduler reaches 25.21 PPL, compared with 22.79 for SOLAR (Table 22).

C.13.2 Untrained Fixed-Policy Controller

This control keeps the SOLAR scheduler architecture, state inputs, and residual action parameterization, but fixes the policy at its random initialization and performs no PPO updates. The base schedule, action-scale warmup, and training setup match SOLAR.

The untrained fixed-policy controller reaches 25.04 PPL. Target-run policy updates reduce this value to 22.79 with SOLAR-online.

Table 22: Controls for state-independent stochasticity and untrained policy initialization. All experiments use the 130M Llama 2 model with AdamW on C4.

Method	**State-conditioned**	**PPO updates**	**Stochastic sampling**	**Action noise**	**Final Eval PPL ↓**
SOLAR	✓	✓	✓	learned	22.79
State-agnostic stochastic scheduler	–	–	✓	selected σ	25.21
Untrained fixed-policy controller	✓	–	✓	fixed init	25.04

The state-agnostic and untrained fixed-policy controls finish at 25.21 and 25.04 PPL, while SOLAR finishes at 22.79.

C.14 Runtime Accounting

Table 23: Steady-state step-time overhead on the 1B AdamW setting with 8×RTX 4090. The post-warmup benchmark window excludes evaluation and checkpointing.

Method	Step Time (s)	Throughput (tok/s)	Overhead Increase (%)
Baseline LRS	1.976	66329.01	0.00
SOLAR (frozen)	1.992	65799.20	0.81
SOLAR (online)	2.001	65499.35	1.27

Table 24: Full-run wall-clock accounting. These measurements include evaluation, checkpointing, state construction, policy inference, action application, PPO updates for the online mode, and any rollback time. No listed run triggered rollback. Dense and Qwen2-MoE runs use 8×RTX 4090; the 3B MoE runs use 32×A800 80GB.

Setting	Baseline (h)	SOLAR-online (h)	Online OH	Frozen OH
1B AdamW	56.75	57.45	1.23%	0.76%
1B Muon	65.34	66.04	1.07%	0.69%
Qwen2-MoE 1B	127.38	129.27	1.48%	–
DeepSeek-V2-style MoE 3B	163.38	165.00	0.99%	–

SOLAR does not add an LLM forward or backward pass. Its incremental work comprises reduction-based state construction over resident tensors, a small rank-0 policy network, one scalar-action

broadcast per controlled group, and periodic PPO updates in online mode. Freezing the residual policy removes the PPO update while retaining state construction and policy inference.

Policy-side complexity. Let G be the number of controlled groups, D the state dimension, H the hidden width, and A the action dimension. The two-hidden-layer policy costs

$$\mathcal{O}\big(DH + H^2 + HA\big)\,.$$

per group and

$$\mathcal{O}\big(G(DH + H^2 + HA)\big)\,.$$

Here $A = 1$ per group. State extraction uses reductions over existing parameters and gradients; PPO backpropagation is amortized because it occurs periodically rather than at every model update.

Step time and full-run wall-clock time. Table 23 isolates the incremental cost in a steady-state benchmark window. It deliberately excludes evaluation and checkpointing so that the step-level mechanism can be measured without cadence effects. Throughput is computed as

$$\text{Throughput} = \frac{B_{\text{global}} \cdot L_{\max}}{t_{\text{step}}}.$$

On 1B AdamW, the measured step-time increases are 1.27% online and 0.81% frozen.

Table 24 times each complete training job from start to finish. It includes evaluation and checkpointing as shared work, as well as SOLAR's checkpoint maintenance and any recovery time. The 1B AdamW full-run overhead is 1.23%, slightly below the 1.27% step-time value because common evaluation and checkpoint work enlarge the full-run denominator. Across the listed complete runs, online overhead ranges from 0.99% to 1.48%.

The 3B MoE, 1B dense, and Qwen2-MoE settings use 155, 219, and 593 controlled groups and show 0.99%, 1.23%, and 1.48% full-run overhead, respectively. The dense implementation groups one trainable tensor at a time, while the MoE counts follow each codebase's module partition.