MOMENTUM AS RESIDUAL-DRIVEN MULTIPLIER CORRECTION FOR DEEP LEARNING OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Momentum-based optimizers are widely used in modern deep learning, yet the relations among momentum, update geometry, and acceleration remain only partially understood. We develop an **A**DMM-**I**nspired **M**omentum (AIM) framework based on residual-penalty variable splitting, which interprets momentum as a multiplier-like correction driven by the splitting residual. AIM recovers the exponential moving average of gradients from an ADMM-style multiplier update and separates three mechanisms that are often intertwined in practical optimizers: the residual penalty determines the update geometry, the multiplier update determines how gradient information is accumulated into momentum, and the approximation of the objective-related θ-subproblem determines the acceleration form. Building on AIM, we propose **R**elativistic **A**daptive gradient **D**escent with **A**ccelerated **R**esidual (RADAR), which combines relativistic adaptive geometry, decoupled residual correction, and gradient-difference momentum filtering to improve the update direction and momentum estimation. We establish stochastic convergence through a variance-perturbed Lyapunov drift analysis. Experiments on supervised vision learning, language modeling, and reinforcement learning show that RADAR achieves consistent improvements over strong adaptive optimizer baselines.

1 INTRODUCTION

Optimization algorithms are a fundamental component of modern deep learning. Among widely used optimizers, SGD with momentum (Polyak, 1964) remains a strong baseline due to its generalization ability, while Adam-type methods (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) have become standard choices because of their robustness and adaptive coordinate-wise scaling. Despite their different update rules, many successful optimizers share several recurring ingredients: a momentum estimator that aggregates gradient information, an update geometry that determines the descent direction, and an acceleration-style correction that refines the parameter update. Understanding how these ingredients interact is important for explaining existing optimizers and designing new ones.

Momentum is commonly viewed as an exponential moving average of gradients, or equivalently as a low-pass filter that suppresses stochastic noise. While this interpretation explains its variance-reduction effect, it does not clarify why momentum takes its specific recursive form or how it should interact with parameter updates. Acceleration methods offer a complementary view: Nesterov momentum (Nesterov, 1983) introduces a forward-looking correction to heavy-ball momentum, while NAdam (Dozat, 2016) combines this idea with adaptive preconditioning. Yet these methods are typically presented through specific update rules, leaving the underlying correction principle implicit.

A complementary line of work explains optimizer updates from the viewpoint of geometry and dynamical constraints. Norm-based analyses relate different optimizers to steepest descent under different geometries (Bernstein & Newhouse, 2024): Adam-type methods can be understood through coordinate-wise adaptive geometry, while matrix-norm optimizers such as Muon (Jordan et al., 2024) are associated with matrix-level steepest descent. From another perspective, RAD (Lyu et al., 2025) introduces relativistic adaptive geometry and conformal symplectic structure into Adam-type updates, aiming to improve long-term training stability through speed-limiting and structure-preserving considerations. These views clarify important aspects of update geometry and training dynamics, but they do not fully explain how momentum estimation, geometric preconditioning, and acceleration-style correction should be coupled in a single optimizer. As a result, momentum is often treated

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.

as a filtering heuristic, geometry as a preconditioner, and acceleration as an additional correction rule. A framework that separates these roles while keeping their interactions explicit would therefore provide both a unified interpretation of existing optimizers and a systematic route for designing new momentum-based algorithms.

In this paper, we develop an **A**DMM-**I**nspired **M**omentum (AIM) framework through residual-penalty variable splitting. Unlike approaches in which the momentum mechanism and the parameter-update rule are designed largely as separate components, AIM derives both from a common ADMM optimization formulation. By introducing an auxiliary descent variable and coupling it with the network parameter through a splitting residual, AIM interprets momentum as a multiplier-like variable updated by a relaxed residual-driven correction.

AIM further separates three components that are often intertwined in practical optimizers. The residual penalty determines the update geometry; the multiplier or momentum-filtering rule determines how gradient information is accumulated; and the approximation of the θ-subproblem determines the acceleration mechanism through the correction of the tentative descent point. Under Euclidean or weighted quadratic geometries, different approximations of the θ-subproblem recover familiar heavy-ball- and Nesterov-type acceleration forms, which can often be expressed equivalently through modified momentum updates. This equivalence, however, need not persist under non-Euclidean geometries such as spectral-norm geometry, where momentum estimation and acceleration become distinct design choices. Together with different residual geometries, these components define a unified design space and provide a systematic route for constructing new optimizer variants.

Building on this framework, we propose **R**elativistic **A**daptive gradient **D**escent with **A**ccelerated **R**esidual (RADAR). RADAR retains the relativistic adaptive preconditioning of RAD while introducing gradient-difference momentum filtering and decoupled residual-correction. These modifications introduce acceleration into the RAD update while preserving its preconditioning geometry.

Our contributions are summarized as follows:

- We propose AIM, an ADMM-inspired residual-penalty splitting framework that jointly connects momentum mechanisms and parameter-update rules. AIM interprets momentum as a multiplier-like variable and derives the exponential moving-average recursion from a relaxed residual-driven multiplier update. The corresponding full correction enforces the KKT stationarity condition with respect to θ, providing an optimization-based interpretation of momentum rather than treating it as an independently postulated heuristic.
- We use AIM to distinguish three optimizer design components: residual geometry, multiplier or momentum filtering, and approximation of the objective-related θ-subproblem. Different residual penalties induce Euclidean, adaptive diagonal, and matrix-level update geometries; different multiplier or filtering rules determine how gradient information is accumulated into momentum; and different approximations of the θ-subproblem determine the acceleration mechanism. This decomposition provides a unified interpretation of momentum-based optimizer families and a systematic design space for constructing new variants.
- We propose RADAR, an AIM-derived extension of RAD. RADAR combines relativistic adaptive preconditioning, gradient-difference momentum filtering, and decoupled residual correction. We establish its stochastic convergence, and experiments on supervised vision learning, language modeling, and reinforcement learning validate its effectiveness.

2 PRELIMINARIES

We briefly recall the **A**lternating **D**irection **M**ethod of **M**ultipliers (ADMM) (Beck, 2017) for the linearly constrained problem

$$\min_{x,z} \ \phi(x) + \varphi(z), \qquad \text{s.t. } Ax + Bz = c, \tag{1}$$

where ϕ and φ are proper closed convex functions. Its augmented Lagrangian is

$$L_\rho(x, z, \lambda) = \phi(x) + \varphi(z) + \langle \lambda, Ax + Bz - c\rangle + \frac{\rho}{2}\|Ax + Bz - c\|^2, \tag{2}$$

where λ is the Lagrange multiplier and $\rho > 0$ is the penalty parameter. The last term is the standard quadratic penalty on the primal residual. ADMM alternates between two primal minimization steps

and a multiplier update:

$$x_{k+1} = \arg\min_x L_\rho(x, z_k, \lambda_k), \tag{3}$$

$$z_{k+1} = \arg\min_z L_\rho(x_{k+1}, z, \lambda_k), \tag{4}$$

$$\lambda_{k+1} = \lambda_k + \tau\rho(Ax_{k+1} + Bz_{k+1} - c), \tag{5}$$

where $\tau > 0$ is the stepsize of the multiplier update. In the standard case $\tau = 1$, combining the multiplier update with the optimality condition of 4 yields the KKT stationarity condition of Problem 1 with respect to z. This classical connection motivates the multiplier construction in AIM. As established in Appendix A.1, combining the AIM multiplier update with the optimality condition of the θ-subproblem analogously yields the KKT stationarity condition with respect to θ, while the resulting multiplier recursion takes the form of an exponential moving average.

3 An ADMM-Inspired Framework for Momentum-Based Optimization

Motivated by the residual-driven multiplier update in classical ADMM, we introduce a residual-penalty variable-splitting framework for momentum-based neural network optimization. By combining the multiplier update with the optimality condition of the parameter subproblem, AIM connects the resulting multiplier recursion to exponential moving-average momentum while tracking the θ-stationarity component under exact subproblem solves. This perspective further separates three mechanisms that are typically intertwined in practical optimizers: the residual penalty determines the update geometry, the multiplier or momentum-filtering rule determines how gradient information is accumulated, and the approximation of the objective-related θ-subproblem determines the acceleration mechanism.

3.1 Residual-Penalty Splitting and Multiplier Momentum

Consider the empirical risk minimization problem as follows:

$$\min_{\theta\in\mathbb{R}^d} \mathcal{L}(\theta) := \frac{1}{N}\sum_{i=1}^{N} l(\theta, \xi_i), \tag{6}$$

where $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ is the objective function, $l(\theta, \xi)$ is the sample-wise loss, and $\theta \in \mathbb{R}^d$ is the trainable parameter. By introducing an auxiliary descent variable y and the equality constraint $y = \theta$, we rewrite equation 6 as

$$\min_{\theta,y} \ \mathcal{L}(\theta), \qquad \text{s.t. } y - \theta = 0. \tag{7}$$

Here, θ carries the original objective, while y is an auxiliary copy introduced to isolate the descent step. During alternating updates, y and θ may differ, and the discrepancy $y - \theta$ is the splitting residual to be controlled.

Motivated by the augmented Lagrangian of ADMM, we define the residual-penalty augmented Lagrangian as follows:

$$\overline{L}_\rho(\theta, y, m) = \mathcal{L}(\theta) + \langle m, y - \theta\rangle + \frac{\rho}{2}\psi(y - \theta), \tag{8}$$

where m is a multiplier-like variable, $\rho > 0$, and ψ is a proper closed convex residual penalty satisfying $\psi(0) = \min_r \psi(r) = 0$. We assume that the displayed subproblems attain solutions and select subgradients satisfying their optimality conditions when an exact solution is invoked. The classical quadratic penalty corresponds to $\psi(r) = \|r\|^2$, which measures the residual in the Euclidean geometry. Allowing a general convex ψ extends the residual geometry beyond the Euclidean quadratic case, thereby inducing a broader class of update geometries.

We refer to this residual-penalty splitting construction as the **A**DMM-**I**nspired **M**omentum (AIM) framework, summarized in Algorithm 1. The y-subproblem contains no objective term and therefore determines the descent geometry induced by ψ, while the θ-subproblem contains $\mathcal{L}(\theta)$ and determines how objective-gradient information corrects the tentative descent point y_{k+1}. The penalty parameters

Algorithm 1 ADMM-Inspired Momentum (AIM) Framework

Require: $\theta_0 = y_0, m_0, \beta_1 \in [0, 1), \rho_1, \rho_2 > 0, T$
1: **for** $k = 0, 1, \ldots, T-1$ **do**
2: $\quad y_{k+1} = \arg\min_y \left\{ \langle m_k, y - \theta_k \rangle + \frac{\rho_1}{2} \psi(y - \theta_k) \right\}$
3: $\quad \theta_{k+1} \approx \arg\min_\theta \left\{ \mathcal{L}(\theta) + \langle m_k, y_{k+1} - \theta \rangle + \frac{\rho_2}{2} \psi(y_{k+1} - \theta) \right\}$
4: $\quad m_{k+1} = m_k + (1 - \beta_1) \frac{\rho_2}{2} d_{k+1}, \quad d_{k+1} \in \partial\psi(y_{k+1} - \theta_{k+1})$
5: **end for**
6: **return** θ_T

ρ_1 and ρ_2 control the violations of the corresponding splitting constraints. The multiplier is updated using the subgradient of ψ evaluated at the splitting residual. As shown in Appendix A.1, a full multiplier correction enforces the θ-stationarity component under an exact subproblem solve, whereas the relaxed update yields EMA tracking. This connection motivates the multiplier construction in AIM and links its residual-driven update to the optimality structure of the original problem.

We next show how the AIM multiplier recovers the standard momentum recursion and how its residual-driven dynamics lead to a filtering interpretation. Assuming that the θ-subproblem is solved exactly, its first-order optimality condition gives

$$0 = \nabla\mathcal{L}(\theta_{k+1}) - m_k - \frac{\rho_2}{2} d_{k+1}, \qquad d_{k+1} \in \partial\psi(y_{k+1} - \theta_{k+1}). \tag{9}$$

Combining this condition with the multiplier update

$$m_{k+1} = m_k + (1 - \beta_1) \frac{\rho_2}{2} d_{k+1} \tag{10}$$

yields $m_{k+1} = \beta_1 m_k + (1 - \beta_1)\nabla\mathcal{L}(\theta_{k+1})$, which is precisely the exponential moving-average momentum recursion. Thus, momentum arises naturally from the residual-driven multiplier update.

Equation 10 shows that the multiplier accumulates the geometry-dependent residual signal d_{k+1} associated with the splitting discrepancy $y_{k+1} - \theta_{k+1}$. Using the conjugate subgradient relation $d \in \partial\psi(r) \Leftrightarrow r \in \partial\psi^*(d)$ together with equation 9, we obtain

$$y_{k+1} - \theta_{k+1} \in \partial\psi^* \left(\frac{2}{\rho_2} \big(\nabla\mathcal{L}(\theta_{k+1}) - m_k \big) \right). \tag{11}$$

Hence, the splitting discrepancy between the auxiliary variable and the parameter is directly related, through the conjugate geometry induced by ψ, to the mismatch between the current gradient and the momentum estimate. Since the multiplier dynamics regulate this splitting discrepancy through residual accumulation, they can equivalently be interpreted as regulating the gradient–momentum mismatch in the corresponding geometry. This provides the bridge from the multiplier view of momentum to the control and filtering interpretation developed below.

3.2 Acceleration from the θ-Subproblem

We next show how acceleration arises from a more refined treatment of the θ-subproblem. For the explicit optimizer, we first form an EMA momentum estimate from the available gradient. Direct assignment then gives momentum descent, while a one-step refinement produces a Nesterov-type corrected direction. These explicit updates use the EMA rule independently of the approximate subproblem solve; they are not literal instances of the residual multiplier update in Algorithm 1.

Specifically, for $\psi(r) = \|r\|^2$, the y-subproblem gives

$$y_{k+1} = \theta_k - \frac{1}{\rho_1} m_k. \tag{12}$$

For the explicit implementation, define $g_k = \nabla\mathcal{L}(\theta_k)$ and $m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k$ before computing the y-step. Taking $\rho_1 = 1/\eta$ yields the tentative momentum descent point $y_{k+1} = \theta_k - \eta m_k$. The exact optimality condition of the θ-subproblem can be written as the implicit fixed-point equation

$$\theta_{k+1} = y_{k+1} - \eta(1 - \beta_1)\big(\nabla\mathcal{L}(\theta_{k+1}) - m_k\big), \qquad \rho_2 = \frac{1}{\eta(1 - \beta_1)}. \tag{13}$$

This equation refines y_{k+1} by a correction proportional to the gradient–momentum mismatch. Since the correction depends on the unknown θ_{k+1}, the exact update is implicit. Applying a one-step fixed-point iteration initialized at θ_k gives the explicit approximation

$$\begin{aligned}\theta_{k+1} &= y_{k+1} - \eta(1-\beta_1)\big(\nabla\mathcal{L}(\theta_k) - m_k\big) \\ &= \underbrace{\theta_k - \eta m_k}_{\text{momentum descent}} \underbrace{-\eta(1-\beta_1)\big(\nabla\mathcal{L}(\theta_k) - m_k\big)}_{\text{residual correction}}.\end{aligned} \tag{14}$$

Theorem 1. *Consider the explicit EMA implementation defined above under the Euclidean geometry $\psi(r) = \|r\|^2$, with $\rho_1 = 1/\eta$ and $\rho_2 = 1/(\eta(1-\beta_1))$. If the θ-subproblem is approximated by the direct assignment $\theta_{k+1} = y_{k+1}$, the update is EMA momentum descent. If instead one fixed-point iteration is applied to the θ-subproblem, the update uses the Nesterov-type direction $\beta_1 m_k + (1-\beta_1)g_k$.*

Detailed derivations for Theorem 1 are provided in Appendix A.2. Theorem 1 describes the explicit EMA implementation. If Algorithm 1 instead applies its residual update after direct assignment, then $m_{k+1} = m_k$; after the stated one-step approximation, it yields $m_{k+1} = \beta_1 m_k + (1-\beta_1)g_k$ and $\theta_{k+1} = \theta_k - \eta m_{k+1}$. The exact splitting identity and its explicit optimizer interpretation must therefore be distinguished.

3.3 Adaptive Geometry from the y-Subproblem

While Section 3.2 shows that the treatment of the θ-subproblem determines the acceleration mechanism, we now turn to the y-subproblem, which determines the update geometry. Since the y-subproblem contains no objective term, the choice of residual penalty ψ directly specifies the geometry of the descent step.

For Adam-type methods, let v_k denote the second-moment estimate available at iteration k:

$$v_k = \beta_2 v_{k-1} + (1-\beta_2)\nabla\mathcal{L}(\theta_k)^2, \qquad Q_k = \mathrm{Diag}(\sqrt{v_k} + \epsilon), \tag{15}$$

where β_2 is the second-moment coefficient. Using the weighted residual penalty $\psi(r) = \|r\|_{Q_k}^2$ yields the adaptive-geometry y-update

$$y_{k+1} = \theta_k - \eta Q_k^{-1} m_k, \qquad \rho_1 = 1/\eta. \tag{16}$$

Thus, the second-moment estimate defines an adaptive residual geometry through Q_k, while leaving the multiplier-like role of m_k unchanged.

Under this geometry, directly setting $\theta_{k+1} = y_{k+1}$ recovers Adam without bias correction, whereas applying one fixed-point iteration to the θ-subproblem gives

$$\begin{aligned}\theta_{k+1} &= y_{k+1} - \eta(1-\beta_1)Q_k^{-1}\big(\nabla\mathcal{L}(\theta_k) - m_k\big) \\ &= \theta_k - \eta Q_k^{-1} m_k - \eta(1-\beta_1)Q_k^{-1}\big(\nabla\mathcal{L}(\theta_k) - m_k\big),\end{aligned} \tag{17}$$

which recovers NAdam without bias correction. Therefore, Adam and NAdam share the same adaptive residual geometry and differ only in the treatment of the θ-subproblem.

The same geometric view extends naturally to matrix-valued parameters. Replacing the adaptive diagonal geometry with spectral-norm geometry yields a matrix-sign descent direction, and under the direct assignment $\Theta_{k+1} = Y_{k+1}$ the resulting update recovers a Muon-type method. Detailed derivations are provided in Appendix A.3.

3.4 A Unified Optimizer Design Space

The preceding analysis reveals two fundamental design axes within AIM. The residual penalty ψ determines the geometry of the tentative update, whereas the treatment of the θ-subproblem determines the acceleration mechanism built upon this geometry. A direct assignment retains only the basic update induced by the chosen geometry, while a refined approximation, such as one fixed-point iteration, introduces an additional residual correction that can give rise to acceleration. The multiplier recursion complements these two choices by specifying the momentum state.

Table 1: Representative optimizer instances within the AIM design space. The residual penalty ψ determines the update geometry, while the approximation of the θ-subproblem determines the associated correction or acceleration mechanism.

Residual geometry	**Direct assignment**	**One fixed-point iteration**	**Alternative approximations**
Euclidean Norm $\psi(\cdot) = \|\cdot\|^2$	SGD-M	Nesterov momentum	Open direction
Weighted Norm $\psi(\cdot) = \|\cdot\|_Q^2$	Adam, RAD, and other adaptive optimizers	NAdam, RADAR, and other accelerated adaptive optimizers	Open direction
Spectral Norm $\psi(\cdot) = \|\cdot\|_2^2$	Muon-type direction	Open direction	Open direction
Infinity Norm $\psi(\cdot) = \|\cdot\|_\infty^2$	Lion-type direction	Open direction	Open direction

Accordingly, a broad class of existing optimizers can be organized as different combinations of update geometry and θ-subproblem treatment, as summarized in Table 1. Beyond providing a unified interpretation, this perspective suggests a systematic principle for designing new optimizers: once the update geometry is specified through ψ, different approximations of the θ-subproblem can be used to construct corresponding correction and acceleration mechanisms. For example, combining spectral-norm or infinity-norm geometry with a fixed-point refinement suggests accelerated variants of Muon- or Lion-type methods. Thus, unexplored combinations in Table 1 define a structured design space for future optimizer development.

4 Relativistic Adaptive Gradient Descent with Accelerated Residual

Building on the AIM framework developed in Section 3, we propose **R**elativistic **A**daptive gradient **D**escent with **A**ccelerated **R**esidual (RADAR). RADAR instantiates the two AIM design axes through a relativistic adaptive geometry and a decoupled residual correction. The former inherits the step-limiting behavior and coordinate-wise adaptivity of RAD (Lyu et al., 2025), while the latter exploits the gradient–momentum mismatch to refine the update direction. In addition, RADAR incorporates gradient-difference filtering into the momentum construction to improve its responsiveness to changes in the gradient trajectory. Together, these components combine RAD's adaptive geometry with the residual-correction principle of AIM and a more responsive momentum estimator.

4.1 Algorithm Design

We adopt the standard adaptive-optimizer indexing convention, yielding an index-shifted practical implementation of the AIM updates.

Relativistic adaptive geometry. Section 3.3 shows that the y-subproblem determines the update geometry. Given a mini-batch $\mathcal{B}_k = \{\xi_{k,i}\}_{i=1}^{\mathcal{B}_k}$, let $g_k = \frac{1}{|\mathcal{B}_k|}\sum_{\xi\in\mathcal{B}_k}\nabla l(\theta_k,\xi)$ be the stochastic gradient estimator of $\nabla\mathcal{L}(\theta_k)$. RADAR updates the second-moment estimate by

$$v_{k+1} = \beta_2 v_k + (1-\beta_2) g_k^2, \tag{18}$$

where all vector operations are element-wise. We define the relativistic adaptive geometry matrix as

$$R_{k+1} := \mathrm{Diag}\big(\sqrt{\delta^2 v_{k+1} + \zeta}\big), \tag{19}$$

where $\delta > 0$ is the speed coefficient and $\zeta \in (0, 1]$ is the symplectic factor. The speed coefficient controls the strength of gradient normalization, and a larger δ imposes stronger coordinate-wise speed limitation. The symplectic factor controls the adaptivity level and is inherited from the relativistic Hamiltonian interpretation of RAD (Lyu et al., 2025).

Algorithm 2 Relativistic Adaptive Gradient Descent with Accelerated Residual (RADAR)

Require: Initial parameter θ_0, learning rate $\eta > 0$, correction coefficient $\ell \geq 0$, momentum coefficients $\beta_1, \beta_2 \in [0, 1)$, filtering coefficient $\gamma \geq 0$, speed coefficient $\delta > 0$, symplectic factor $\zeta \in (0, 1]$, maximum iteration number T

1: Initialize $m_0 = 0$, $v_0 = 0$, and $g_{-1} = 0$
2: **for** $k = 0, 1, \ldots, T-1$ **do**
3: Draw a mini-batch $\mathcal{B}_k$ and compute $g_k = \frac{1}{|\mathcal{B}_k|} \sum_{\xi \in \mathcal{B}_k} \nabla l(\theta_k, \xi)$
4: $m_{k+1} = \beta_1 m_k + (1 - \beta_1) g_k + \gamma(g_k - g_{k-1})$
5: $v_{k+1} = \beta_2 v_k + (1 - \beta_2) g_k^2$
6: $R_{k+1} = \mathrm{Diag}\big(\sqrt{\delta^2 v_{k+1} + \zeta}\big)$
7: $\theta_{k+1} = \theta_k - \eta R_{k+1}^{-1} m_{k+1} - \ell R_{k+1}^{-1}(g_k - m_{k+1})$
8: **end for**
9: **return** θ_T

Given the filtered momentum estimate m_{k+1}, under this geometry, the AIM y-subproblem gives the tentative descent point as follows:

$$y_{k+1} = \theta_k - \eta R_{k+1}^{-1} m_{k+1}. \tag{20}$$

Decoupled residual correction. The AIM framework shows that the θ-subproblem corrects the tentative descent point using the gradient–momentum mismatch. Under relativistic adaptive geometry equation 19, this correction takes the preconditioned form $R_{k+1}^{-1}(m_{k+1} - g_k)$. Instead of tying its scale to the main learning rate and the momentum coefficient as in equation 14, we introduce an independent correction coefficient ℓ:

$$\begin{aligned} \theta_{k+1} &= y_{k+1} + \ell R_{k+1}^{-1}(m_{k+1} - g_k) \\ &= \underbrace{\theta_k - \eta R_{k+1}^{-1} m_{k+1}}_{\text{relativistic momentum descent}} \underbrace{- \ell R_{k+1}^{-1}(g_k - m_{k+1})}_{\text{decoupled residual correction}}. \end{aligned} \tag{21}$$

When $\ell = 0$, RADAR reduces to a relativistic adaptive momentum update. When $\ell > 0$, the residual term provides an acceleration-like correction to the parameter update.

Gradient-difference momentum filtering. The third component of RADAR specifies the multiplier-like momentum state. Beyond the standard exponential moving average, we incorporate recent gradient variation through

$$m_{k+1} = \beta_1 m_k + (1 - \beta_1) g_k + \gamma(g_k - g_{k-1}), \tag{22}$$

where $\gamma \geq 0$ controls the contribution of recent gradient variation. When $\gamma = 0$, equation 22 reduces to the standard exponential moving-average momentum update.

Following the frequency-domain perspective that interprets momentum methods as filters acting on stochastic gradients (Li et al., 2025), we further characterize the fixed-coefficient recursion in equation 22. It can be equivalently written as $m_{k+1} = \beta_1 m_k + (1 - \beta_1 + \gamma) g_k - \gamma g_{k-1}$.

Defining $\widetilde{m}_k = m_{k+1}$, the corresponding transfer function is

$$H(z) = \frac{\widetilde{M}(z)}{G(z)} = \frac{1 - \beta_1 + \gamma - \gamma z^{-1}}{1 - \beta_1 z^{-1}}. \tag{23}$$

The recursion therefore contains one pole and one zero. The exponential moving-average branch provides recursive low-pass behavior, whereas the gradient-difference branch emphasizes recent changes in the stochastic gradient.

Combining relativistic adaptive geometry, decoupled residual correction, and gradient-difference momentum filtering gives Algorithm 2. In Appendix B.4, we further optimize the implementation of RADAR by algebraically reparameterizing the gradient-difference momentum, which eliminates the need to explicitly store the previous gradient.

5 NUMERICAL EXPERIMENTS

We evaluate RADAR on supervised vision learning, language modeling, and reinforcement learning to assess whether the AIM-derived optimizer RADAR provides consistent gains across training regimes with different model architectures, data modalities, and optimization dynamics. We compare RADAR with RAD, Adam, AdamW, NAdam,Adan(Xie et al., 2024),AdaBelief(Zhuang et al., 2020) and Lion(Chen et al., 2023). For fair comparison, we conduct learning-rate searches with the same task-specific tuning budget for all optimizers on each supervised learning task. After selecting the optimal learning rate for each optimizer, we rerun each configuration using five independent random seeds, except for the reinforcement-learning tasks, for which ten seeds are used. The detailed experimental settings are reported in Appendix B. Except for the Qwen experiment using $4\times$A100 GPUs,all reported experiments are conducted on a single NVIDIA RTX 5090 GPU.

5.1 EXPERIMENTAL SETUP

Supervised vision learning. We evaluate the image classification performance on CIFAR-10 and CIFAR-100 (Krizhevsky, 2009). We train a Vision Transformer (ViT, 6.3M parameters) (Dosovitskiy et al., 2021) on CIFAR-10 and a ResNet-50 (He et al., 2016) on CIFAR-100 with standard data augmentation, following the training configuration of ROOT (He et al., 2025). Each optimizer is trained for 100 epochs. We report the mean and standard deviation of test loss.

Language modeling. We evaluate GPT-2 (124M) (Radford et al., 2019) in both pre-training and fine-tuning settings. For pre-training, we pretrain GPT-2 on WikiText-103 (Merity et al., 2017) for approximately 3 epochs. For fine-tuning, we fine-tune GPT-2 on WikiText-2 (Merity et al., 2017) for 3 epochs. To assess the scalability of the compared optimizers, we additionally pretrain Qwen2.5-0.5B (Yang et al., 2024) on FineWeb-Edu (Penedo et al., 2024) with a total training budget of 1B tokens. We report test perplexity for the GPT-2 settings and validation perplexity for Qwen, with means and standard deviations over the reported seeds. Detailed language-modeling settings are provided in Appendix B.3.

Reinforcement learning. We evaluate long-horizon continuous control on MuJoCo benchmarks (Todorov et al., 2012; Brockman et al., 2016). In the main text, we report Soft Actor-Critic (SAC) (Haarnoja et al., 2018) results on HalfCheetah-v4 and Walker2d-v4. All optimizers are plugged into the same GOPS (Wang et al., 2023) implementation with identical training and evaluation protocols. We report the total average return (TAR) and standard deviation over 10 random seeds.

5.2 MAIN RESULTS

Table 2 summarizes the main results across supervised vision learning, language modeling, and reinforcement learning. RADAR achieves the best mean performance in all seven evaluated settings, spanning different model architectures, data modalities, and optimization objectives, showing consistent improvements in mean performance across diverse learning problems.

For supervised vision learning, RADAR achieves the lowest mean test loss on both CIFAR-10 with ViT and CIFAR-100 with ResNet-50, obtaining 0.8011 ± 0.0112 and 1.6190 ± 0.0088, respectively, and consistently performing well across Transformer and convolutional architectures.

The gains are more pronounced in language-model pre-training. On WikiText-103 with GPT-2, RADAR achieves the best test perplexity of 22.4963 ± 0.0933, compared with 23.0514 ± 0.1909 for Adan. On FineWeb-Edu with Qwen2.5-0.5B, RADAR again achieves the best validation perplexity of 30.3029 ± 0.2812, followed by RAD at 30.8392 ± 0.7031. Figure 1 further shows that RADAR maintains the lowest late-stage training loss on both tasks, suggesting consistent gains across model scales.

For GPT-2 fine-tuning on WikiText-2, RADAR achieves the best mean perplexity of 21.5358 ± 0.0082, slightly outperforming NAdam at 21.5513 ± 0.0054. Additional language-modeling results are provided in Appendix B.3.

Table 2: Main results across supervised vision learning, language modeling, and reinforcement learning. Results are reported as mean ± standard deviation. Best results are in bold and second-best results are underlined.

Task	Metric	RADAR	RAD	Adam	AdamW	NAdam	Adan	AdaBelief	Lion
CIFAR-10, ViT(6.3M)	Test Loss ↓	**0.8011 ±0.0112**	0.8307 ±0.0051	0.8240 ±0.0078	0.8339 ±0.0039	0.8200 ±0.0107	0.8029 ±0.0086	0.8322 ±0.0078	0.8063 ±0.0089
CIFAR-100, ResNet-50	Test Loss ↓	**1.6190 ±0.0088**	1.6787 ±0.0108	1.6244 ±0.0100	1.6771 ±0.0191	1.6326 ±0.0122	1.6338 ±0.0068	1.6831 ±0.0086	1.7078 ±0.0152
WikiText-103, GPT-2 Small	Test PPL ↓	**22.4963 ±0.0933**	26.0321 ±0.0331	26.0733 ±0.0321	26.0281 ±0.0433	23.9754 ±0.2368	23.0514 ±0.1909	25.5079 ±0.0738	26.1346 ±0.2068
FineWeb-Edu, Qwen2.5-0.5B	Valid PPL ↓	**30.3029 ±0.2812**	30.8392 ±0.7031	31.7604 ±0.9650	31.3155 ±0.8857	49.1373 ±2.1917	133.2456 ±15.2367	31.0569 ±0.6742	331.9066 ±50.1679
WikiText-2 SFT, GPT-2 Small	Test PPL ↓	**21.5358 ±0.0082**	21.5744 ±0.0133	21.5768 ±0.0132	21.5765 ±0.0129	21.5513 ±0.0054	21.6465 ±0.0092	21.5578 ±0.0169	21.7824 ±0.0412
HalfCheetah-v4, SAC	TAR ↑	**9569 ±926**	8996 ±1025	8557 ±939	8828 ±790	8582 ±1493	8127 ±559	9194 ±828	8251 ±828
Walker2d-v4, SAC	TAR ↑	**3580 ±282**	2687 ±932	2798 ±1117	3196 ±1124	3362 ±776	2561 ±1318	2729 ±980	2877 ±1025

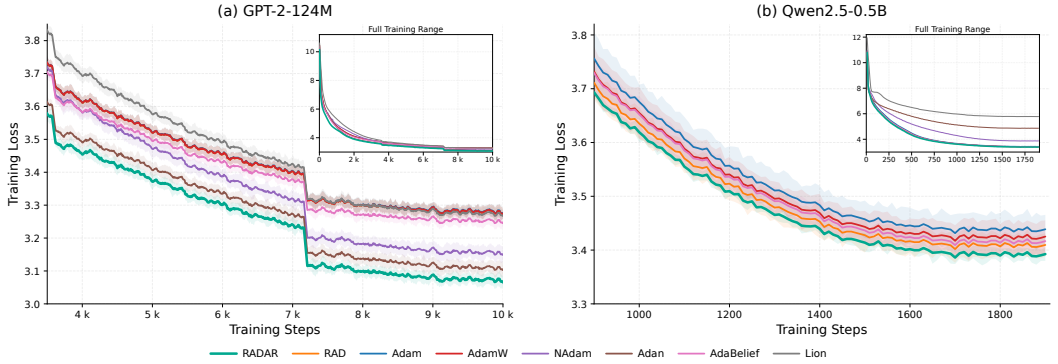

Figure 1: Training-loss curves of different optimizers on language-model pre-training tasks.

RADAR also performs consistently well in reinforcement learning. With SAC, it achieves the highest average return on HalfCheetah-v4 (9569 ± 926) and Walker2d-v4 (3580 ± 282), compared with 9194 ± 828 for AdaBelief and 3362 ± 776 for NAdam, respectively.

Overall, RADAR achieves the best mean performance across all seven evaluated settings, with larger improvements in language-model pre-training and reinforcement learning and smaller but consistent improvements in vision learning and language-model fine-tuning.

6 Conclusion

In this paper, we propose the AIM framework, which interprets momentum as a residual-driven multiplier-like correction and separates update geometry from acceleration form. Based on this view, we develop RADAR, a new optimizer that combines relativistic adaptive geometry, decoupled residual correction, and gradient-difference momentum filtering. We establish stochastic convergence, and experiments on vision, language modeling, and reinforcement learning tasks demonstrate the effectiveness of RADAR. Future work will continue to explore optimizer design within the AIM framework, relax the current analytical assumptions to establish more general convergence guarantees, and further investigate the insights that the AIM perspective provides for understanding existing and future optimization methods.

AI USE STATEMENT

In this work, we used generative AI tools for editing and debugging experimental code, language polishing of the manuscript, and formatting references. Generative AI tools were not used to develop the research ideas, theoretical framework, mathematical derivations, or experimental methodology of this work.

All AI-assisted code and manuscript edits were reviewed and verified by the authors. We take responsibility for the final content of this work, including text and artifacts produced with the aid of generative AI.

REFERENCES

Amir Beck. *First-Order Methods in Optimization*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017. doi: 10.1137/1.9781611974997.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. In *OPT 2024: Optimization for Machine Learning (NeurIPS Workshop)*, pp. 1–19, 2024. URL https://openreview.net/forum?id=ux18f5nOpD.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016. URL https://arxiv.org/abs/1606.01540.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49205–49233, 2023. doi: 10.52202/075280-2140.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Timothy Dozat. Incorporating nesterov momentum into adam. In *International Conference on Learning Representations Workshop Track*, pp. 1–4, 2016. URL https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Wei He, Kai Han, Hang Zhou, Hanting Chen, Zhicheng Liu, Xinghao Chen, and Yunhe Wang. Root: Robust orthogonalized optimizer for neural network training. *arXiv preprint arXiv:2511.20626*, 2025.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. Blog post, 2024. URL https://kellerjordan.github.io/posts/muon/. Accessed: 2026-04-29.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pp. 1–15, 2015. URL https://arxiv.org/abs/1412.6980.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Xianliang Li, Jun Luo, Zhiwei Zheng, Hanxiao Wang, Li Luo, Lingkun Wen, Linlong Wu, and Sheng Xu. On the performance analysis of momentum method: A frequency domain perspective. In *International Conference on Learning Representations*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, pp. 1–18, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yao Lyu, Xiangteng Zhang, Shengbo Eben Li, Jingliang Duan, Letian Tao, Qing Xu, Lei He, and Keqiang Li. Conformal symplectic optimization for stable reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):11049–11063, 2025. doi: 10.1109/TNNLS.2024.3511670.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, pp. 1–15, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Yurii E. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, 2024.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. doi: 10.1016/0041-5553(64)90137-5.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Wenxuan Wang, Yuhang Zhang, Jiaxin Gao, Yuxuan Jiang, Yujie Yang, Zhilong Zheng, Wenjun Zou, Jie Li, Congsheng Zhang, Wenhan Cao, Genjin Xie, Jingliang Duan, and Shengbo Eben Li. GOPS: A general optimal control problem solver for autonomous driving and industrial control applications. *Communications in Transportation Research*, 3:100096, 2023. doi: 10.1016/j.commtr.2023.100096.

Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9508–9520, 2024. doi: 10.1109/TPAMI.2024.3423382.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C. Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18795–18806, 2020.

A Technical Proofs and Derivations

A.1 Derivation of Multiplier Update Rules

We first explain the multiplier update rule in the classical ADMM update equation 5. The optimality condition of the z-subproblem equation 4 gives

$$0 \in \partial\varphi(z_{k+1}) + B^\top \lambda_k + \rho B^\top (Ax_{k+1} + Bz_{k+1} - c). \tag{24}$$

When $\tau = 1$, the multiplier update equation 5 becomes

$$\lambda_{k+1} = \lambda_k + \rho(Ax_{k+1} + Bz_{k+1} - c). \tag{25}$$

Combining the above two relations yields

$$0 \in \partial\varphi(z_{k+1}) + B^\top \lambda_{k+1}. \tag{26}$$

Therefore, the updated pair (z_{k+1}, λ_{k+1}) satisfies the stationarity condition of Problem equation 1 with respect to z. This shows that the ADMM multiplier update can be viewed as a residual-driven dual correction toward the KKT condition.

We next derive the multiplier update rule in AIM. The optimality condition of the θ-subproblem gives

$$0 \in \nabla\mathcal{L}(\theta_{k+1}) - m_k - \frac{\rho_2}{2}\partial\psi(y_{k+1} - \theta_{k+1}). \tag{27}$$

Equivalently, there exists $d_{k+1} \in \partial\psi(y_{k+1} - \theta_{k+1})$ such that

$$\nabla\mathcal{L}(\theta_{k+1}) - m_k - \frac{\rho_2}{2} d_{k+1} = 0. \tag{28}$$

For the splitting problem equation 7, the stationarity condition with respect to θ is

$$\nabla\mathcal{L}(\theta) - m = 0. \tag{29}$$

Thus, a full residual-driven multiplier correction would update

$$m_{k+1} = m_k + \frac{\rho_2}{2} d_{k+1}, \tag{30}$$

which, together with equation 28, gives $m_{k+1} = \nabla\mathcal{L}(\theta_{k+1})$ and hence satisfies the above stationarity condition at θ_{k+1}.

In AIM, we use a relaxed multiplier update with stepsize $1 - \beta_1$:

$$m_{k+1} = m_k + (1 - \beta_1)\frac{\rho_2}{2} d_{k+1}. \tag{31}$$

Substituting equation 28 into equation 31 yields

$$m_{k+1} = m_k + (1 - \beta_1)\big(\nabla\mathcal{L}(\theta_{k+1}) - m_k\big) = \beta_1 m_k + (1 - \beta_1)\nabla\mathcal{L}(\theta_{k+1}). \tag{32}$$

Therefore, the AIM multiplier update recovers the standard exponential moving average form of momentum from a relaxed residual-driven multiplier correction.

A.2 Proof of Theorem 1

Proof. We prove the result by eliminating the auxiliary descent variable y. Under the Euclidean residual penalty $\psi(r) = \|r\|^2$, the y-subproblem in Algorithm 1 becomes

$$y_{k+1} = \arg\min_y \left\{ \langle m_k, y - \theta_k \rangle + \frac{\rho_1}{2}\|y - \theta_k\|^2 \right\}.$$

Its first-order optimality condition is

$$m_k + \rho_1(y_{k+1} - \theta_k) = 0,$$

and hence

$$y_{k+1} = \theta_k - \frac{1}{\rho_1} m_k. \tag{33}$$

Taking $\rho_1 = 1/\eta$, we obtain

$$y_{k+1} = \theta_k - \eta m_k. \tag{34}$$

Thus, y_{k+1} is the tentative momentum descent point.

We first consider the Nesterov-type approximation of the θ-subproblem used in equation 14:

$$\theta_{k+1} = y_{k+1} - \eta(1-\beta_1)\big(\nabla\mathcal{L}(\theta_k) - m_k\big). \tag{35}$$

Substituting equation 34 into equation 35 gives

$$\theta_{k+1} = \theta_k - \eta m_k - \eta(1-\beta_1)\big(\nabla\mathcal{L}(\theta_k) - m_k\big). \tag{36}$$

Equivalently, by defining the corrected momentum direction as follow:

$$\widetilde{m}_k := m_k + (1-\beta_1)\big(\nabla\mathcal{L}(\theta_k) - m_k\big),$$

we can write

$$\theta_{k+1} = \theta_k - \eta\widetilde{m}_k. \tag{37}$$

Therefore, compared with the pure momentum descent direction m_k, the update direction is corrected by the gradient–momentum mismatch $\nabla\mathcal{L}(\theta_k) - m_k$. This is the Nesterov-type correction induced by the one-step approximation of the θ-subproblem in equation 14. Hence, the resulting update recovers a Nesterov-type momentum method up to the change of variables

$$\widetilde{m}_k = m_k + (1-\beta_1)\big(\nabla\mathcal{L}(\theta_k) - m_k\big).$$

We next consider the simpler approximation as follows:

$$\theta_{k+1} = y_{k+1}.$$

Combining it with equation 34 gives

$$\theta_{k+1} = \theta_k - \eta m_k. \tag{38}$$

With the explicitly maintained EMA $m_k = \beta_1 m_{k-1} + (1-\beta_1)g_k$, this is momentum descent. For the refined update, $\widetilde{m}_k = \beta_1 m_k + (1-\beta_1)g_k$. The EMA identity derived for an exact θ-subproblem cannot be substituted into Algorithm 1 after replacing that solve by either approximation. In particular, direct assignment makes its actual residual multiplier update vanish. □

A.3 Details for Adaptive and Matrix-Norm Extensions

We first justify the adaptive update in equation 16. With the weighted residual penalty as follows:

$$\psi(r) = \|r\|_{Q_k}^2, \qquad Q_k = \mathrm{Diag}(\sqrt{v_k} + \epsilon),$$

the y-subproblem becomes

$$y_{k+1} = \arg\min_y \left\{ \langle m_k, y - \theta_k\rangle + \frac{\rho_1}{2}\|y - \theta_k\|_{Q_k}^2 \right\}.$$

Its first-order optimality condition is

$$m_k + \rho_1 Q_k(y_{k+1} - \theta_k) = 0,$$

which gives

$$y_{k+1} = \theta_k - \frac{1}{\rho_1} Q_k^{-1} m_k.$$

Taking $\rho_1 = 1/\eta$ yields

$$y_{k+1} = \theta_k - \eta Q_k^{-1} m_k.$$

This proves the adaptive preconditioned update used in equation 16. The direct approximation $\theta_{k+1} = y_{k+1}$ gives the Adam-type update, while the Nesterov-type approximation in equation 17 gives the NAdam-type update.

We next discuss the matrix-norm case. Let Θ, Y, M denote matrix-valued counterparts of θ, y, m. Under a spectral-norm geometry, the y-step can be interpreted as a normalized steepest-descent step for the local linear model, i.e.,

$$\min_{\Delta} \langle M_k, \Delta\rangle \qquad \text{s.t.} \quad \|\Delta\|_2 \le \eta.$$

Using the duality between the spectral norm and the nuclear norm, an optimal normalized descent direction is given by the negative polar factor of M_k. If

$$M_k = U\Sigma V^\top$$

is the singular value decomposition of M_k, define

$$\mathrm{Orthogonal}(M_k) := UV^\top.$$

Then the matrix-valued descent step is

$$Y_{k+1} = \Theta_k - \eta\,\mathrm{Orthogonal}(M_k).$$

With the direct approximation $\Theta_{k+1} = Y_{k+1}$, we obtain

$$\Theta_{k+1} = \Theta_k - \eta\,\mathrm{Orthogonal}(M_k),$$

which is the Muon-type matrix update.

A.4 Convergence Analysis

We present the convergence analysis of RADAR in Algorithm 2. The proof uses a variance-perturbed Lyapunov drift argument that jointly controls the objective value, the momentum residual, and the successive parameter displacement.

Assumption 1 (Smoothness and lower boundedness)**.** *For any data sample ξ, the sample-wise loss $l(\cdot,\xi)$ is L-smooth. Moreover, $\mathcal{L}$ is lower bounded by $\mathcal{L}_*$.*

Assumption 2 (Unbiased stochastic gradient and bounded variance)**.** *Let g_k be the mini-batch stochastic gradient, where the samples are drawn independently. We assume that the single-sample stochastic gradient is unbiased and has bounded variance:*

$$\mathbb{E}[\nabla l(\theta,\xi)] = \nabla\mathcal{L}(\theta), \qquad \mathbb{E}\left\|\nabla l(\theta,\xi) - \nabla\mathcal{L}(\theta)\right\|^2 \le \sigma^2, \quad \forall\theta.$$

Assumption 3 (Bounded adaptive geometry)**.** *Let R_k be the relativistic adaptive geometry matrix in RADAR as defined in equation 19. There exist constants $0 < \nu_{\min} \le \nu_{\max} < \infty$ such that*

$$\nu_{\min} I \preceq R_k^{-1} \preceq \nu_{\max} I, \quad \forall k. \tag{39}$$

We define $\varepsilon_k := \nabla\mathcal{L}(\theta_k) - g_k$, $r_{k+1} := \nabla\mathcal{L}(\theta_k) - m_{k+1}$ and consider the Lyapunov function as follows:

$$V_k := \mathcal{L}(\theta_k) + c_1\|r_k\|^2 + c_2\|\theta_k - \theta_{k-1}\|^2, \tag{40}$$

where $c_1, c_2 > 0$ are specified in Appendix A.5.

Lemma 1 (Lyapunov drift bound)**.** *Suppose Assumptions 1–3 hold and $0 < \beta_1 < 1$, $0 < \tau < \beta_1^{-2} - 1$. If the parameters $\eta, \ell, \beta_1, \gamma$ are chosen in the stable regime specified in Appendix A.5, then there exist constants $d_1, d_2, C_v > 0$ such that*

$$\mathbb{E}V_{k+1} - \mathbb{E}V_k \le -d_1\mathbb{E}\|r_{k+1}\|^2 - d_2\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 + C_v\sigma^2\left(\frac{1}{|\mathcal{B}_k|} + \frac{1}{|\mathcal{B}_{k-1}|}\right), \tag{41}$$

for $k \ge 1$, where $d_1, d_2 > 0$ are the descent coefficients for the momentum residual and the stationarity measure, while $C_v > 0$ quantifies the amplification of stochastic-gradient variance.

Lemma 1 is a variance-perturbed Lyapunov drift bound. It reduces to a standard Lyapunov descent condition in the full-gradient case and yields a variance-controlled stationary neighborhood in the stochastic case. We define $R(T) = \frac{1}{T-1}\sum_{k=1}^{T-1}\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2$, and give its bound as follows.

Theorem 2 (Sublinear convergence of RADAR)**.** *Under the conditions of Lemma 1, the sequence generated by Algorithm 2 satisfies, for $T \ge 2$,*

$$R(T) \le \frac{\mathbb{E}V_1 - \mathcal{L}_*}{d_2(T-1)} + \frac{C_v\sigma^2}{d_2(T-1)}\sum_{k=1}^{T-1}\left(\frac{1}{|\mathcal{B}_k|} + \frac{1}{|\mathcal{B}_{k-1}|}\right). \tag{42}$$

Therefore, with a fixed mini-batch size, RADAR converges to a variance-controlled stationary neighborhood. If full gradients are used or if the batch-size schedule makes the accumulated variance term uniformly bounded, RADAR achieves an $O(1/T)$ average stationarity bound. The proofs of Lemma 1 and Theorem 2 are provided in Appendix A.5.

A.5 Proofs for the Convergence Analysis of RADAR

A.5.1 Technical estimates

Recall that

$$\varepsilon_k = \nabla\mathcal{L}(\theta_k) - g_k, \qquad r_{k+1} = \nabla\mathcal{L}(\theta_k) - m_{k+1}.$$

We first record a standard mini-batch variance bound. Under Assumption 2, conditioned on θ_k, we have

$$\mathbb{E}[g_k \mid \theta_k] = \mathbb{E}\left[\frac{1}{|\mathcal{B}_k|}\sum_{b=1}^{B_k}\nabla l(\theta_k, \xi_{k,b}) \mid \theta_k\right] = \nabla\mathcal{L}(\theta_k),$$

and

$$g_k - \nabla\mathcal{L}(\theta_k) = \frac{1}{|\mathcal{B}_k|}\sum_{b=1}^{B_k}\left(\nabla l(\theta_k, \xi_{k,b}) - \nabla\mathcal{L}(\theta_k)\right).$$

Since the mini-batch samples are independent, the cross terms vanish. Therefore,

$$\mathbb{E}\|\varepsilon_k\|^2 = \mathbb{E}\|g_k - \nabla\mathcal{L}(\theta_k)\|^2 = \frac{1}{|\mathcal{B}_k|^2}\sum_{b=1}^{B_k}\mathbb{E}\,\|\nabla l(\theta_k, \xi_{k,b}) - \nabla\mathcal{L}(\theta_k)\|^2 \le \frac{\sigma^2}{|\mathcal{B}_k|}. \tag{43}$$

We next derive two estimates used in the Lyapunov analysis. From the gradient-difference momentum filtering update, i.e.,

$$m_{k+1} = \beta_1 m_k + (1-\beta_1)g_k + \gamma(g_k - g_{k-1}),$$

we have

$$\begin{aligned} r_{k+1} &= \nabla\mathcal{L}(\theta_k) - \beta_1 m_k - (1-\beta_1+\gamma)g_k + \gamma g_{k-1} \\ &= \varepsilon_k - \beta_1\varepsilon_{k-1} + \beta_1 r_k + (\beta_1-\gamma)(g_k - g_{k-1}). \end{aligned} \tag{44}$$

Define

$$a_k := \varepsilon_k - \beta_1\varepsilon_{k-1}, \qquad b_k := \beta_1 r_k, \qquad c_k := (\beta_1-\gamma)(g_k - g_{k-1}).$$

Then $r_{k+1} = a_k + b_k + c_k$. By Young's inequality, i.e.,

$$\|x+y\|^2 \le (1+\tau)\|x\|^2 + \left(1+\frac{1}{\tau}\right)\|y\|^2, \qquad \tau > 0.$$

Taking $x = b_k$, $y = a_k + c_k$, and using $\|a_k + c_k\|^2 \le 2\|a_k\|^2 + 2\|c_k\|^2$, it follows that

$$\mathbb{E}\|r_{k+1}\|^2 \le 2(1+\frac{1}{\tau})\mathbb{E}\|\varepsilon_k - \beta_1\varepsilon_{k-1}\|^2 + (1+\tau)\beta_1^2\mathbb{E}\|r_k\|^2 + 2(1+\frac{1}{\tau})(\beta_1-\gamma)^2\mathbb{E}\|g_k - g_{k-1}\|^2. \tag{45}$$

For the first term on the right-hand side of equation 45, we have

$$\mathbb{E}\|\varepsilon_k - \beta_1\varepsilon_{k-1}\|^2 = \mathbb{E}\|\varepsilon_k\|^2 + \beta_1^2\mathbb{E}\|\varepsilon_{k-1}\|^2 \le \frac{\sigma^2}{|\mathcal{B}_k|} + \frac{\sigma^2}{|\mathcal{B}_{k-1}|}, \tag{46}$$

where the first equality holds because $\mathbb{E}\langle\varepsilon_k, \varepsilon_{k-1}\rangle = 0$, and the last inequality follows from equation 43 and $\beta_1 \in [0, 1)$.

For the third term on the right-hand side of equation 45, since

$$g_k = \nabla\mathcal{L}(\theta_k) - \varepsilon_k,$$

we have

$$g_k - g_{k-1} = \nabla\mathcal{L}(\theta_k) - \nabla\mathcal{L}(\theta_{k-1}) - \varepsilon_k + \varepsilon_{k-1}.$$

Using $\|a+b+c\|^2 \le 3\|a\|^2 + 3\|b\|^2 + 3\|c\|^2$, the smoothness of $\mathcal{L}$, and equation 43, we obtain

$$\begin{aligned} \mathbb{E}\|g_k - g_{k-1}\|^2 &\le 3\mathbb{E}\|\nabla\mathcal{L}(\theta_k) - \nabla\mathcal{L}(\theta_{k-1})\|^2 + 3\mathbb{E}\|\varepsilon_k\|^2 + 3\mathbb{E}\|\varepsilon_{k-1}\|^2 \\ &\le 3L^2\mathbb{E}\|\theta_k - \theta_{k-1}\|^2 + 3\frac{\sigma^2}{|\mathcal{B}_k|} + 3\frac{\sigma^2}{|\mathcal{B}_{k-1}|}. \end{aligned} \tag{47}$$

Combining equation 45, equation 46, and equation 47, we obtain

$$\begin{aligned}\mathbb{E}\|r_{k+1}\|^2 \leq &(1+\tau)\beta_1^2\mathbb{E}\|r_k\|^2 + 6(1+\frac{1}{\tau})(\beta_1-\gamma)^2L^2\mathbb{E}\|\theta_k-\theta_{k-1}\|^2 \\ &+ \left[2(1+\frac{1}{\tau}) + 6(1+\frac{1}{\tau})(\beta_1-\gamma)^2\right]\sigma^2\left(\frac{1}{|\mathcal{B}_k|}+\frac{1}{|\mathcal{B}_{k-1}|}\right),\end{aligned} \tag{48}$$

Rearranging equation 48 yields

$$\begin{aligned}\mathbb{E}\|r_{k+1}\|^2 - \mathbb{E}\|r_k\|^2 \leq &- C_r\mathbb{E}\|r_{k+1}\|^2 \\ &+ C_\theta\mathbb{E}\|\theta_k-\theta_{k-1}\|^2 + C_\varepsilon\sigma^2\left(\frac{1}{|\mathcal{B}_k|}+\frac{1}{|\mathcal{B}_{k-1}|}\right).\end{aligned} \tag{49}$$

where the constants are chosen as

$$C_r = \frac{1}{(1+\tau)\beta_1^2} - 1, \qquad C_\theta = \frac{6}{\tau\beta_1^2}(\beta_1-\gamma)^2L^2, \qquad C_\varepsilon = \frac{2}{\tau\beta_1^2}\left[1+3(\beta_1-\gamma)^2\right]. \tag{50}$$

It remains to bound the successive parameter displacement. From Algorithm 2, we have

$$\begin{aligned}\theta_{k+1}-\theta_k &= -\eta R_{k+1}^{-1}m_{k+1} - \ell R_{k+1}^{-1}(g_k - m_{k+1}) \\ &= (\eta-\ell)R_{k+1}^{-1}r_{k+1} + \ell R_{k+1}^{-1}\varepsilon_k - \eta R_{k+1}^{-1}\nabla\mathcal{L}(\theta_k).\end{aligned} \tag{51}$$

By Assumption 3, we have

$$\|x\|_{R_k^{-1}}^2 \leq \nu_{\max}\|x\|^2, \qquad \|R_k^{-1}x\|^2 \leq \nu_{\max}^2\|x\|^2.$$

Using Assumptions 2 and 3, we obtain

$$\mathbb{E}\|\theta_{k+1}-\theta_k\|^2 \leq 3(\eta-\ell)^2\nu_{\max}^2\mathbb{E}\|r_{k+1}\|^2 + 3\eta^2\nu_{\max}^2\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 + 3\ell^2\nu_{\max}^2\frac{\sigma^2}{|\mathcal{B}_k|}. \tag{52}$$

A.5.2 Proof of Lemma 1

Proof. By the L-smoothness of $\mathcal{L}$, we have

$$\mathcal{L}(\theta_{k+1}) \leq \mathcal{L}(\theta_k) + \langle\nabla\mathcal{L}(\theta_k), \theta_{k+1}-\theta_k\rangle + \frac{L}{2}\|\theta_{k+1}-\theta_k\|^2. \tag{53}$$

Substituting equation 51 into equation 53 gives

$$\begin{aligned}\mathcal{L}(\theta_{k+1}) \leq &\mathcal{L}(\theta_k) - \eta\|\nabla\mathcal{L}(\theta_k)\|_{R_{k+1}^{-1}}^2 \\ &+ (\eta-\ell)\langle\nabla\mathcal{L}(\theta_k), R_{k+1}^{-1}r_{k+1}\rangle + \ell\langle\nabla\mathcal{L}(\theta_k), R_{k+1}^{-1}\varepsilon_k\rangle \\ &+ \frac{L}{2}\|\theta_{k+1}-\theta_k\|^2.\end{aligned} \tag{54}$$

Using Young's inequality in the weighted norm and $\eta \geq l$, i.e.,

$$\langle a, R_{k+1}^{-1}b\rangle \leq \frac{1}{2}\|a\|_{R_{k+1}^{-1}}^2 + \frac{1}{2}\|b\|_{R_{k+1}^{-1}}^2,$$

we have

$$(\eta-\ell)\langle\nabla\mathcal{L}(\theta_k), R_{k+1}^{-1}r_{k+1}\rangle \leq \frac{\eta-\ell}{2}\|\nabla\mathcal{L}(\theta_k)\|_{R_{k+1}^{-1}}^2 + \frac{\eta-\ell}{2}\|r_{k+1}\|_{R_{k+1}^{-1}}^2,$$

and

$$\ell\langle\nabla\mathcal{L}(\theta_k), R_{k+1}^{-1}\varepsilon_k\rangle \leq \frac{\ell}{2}\|\nabla\mathcal{L}(\theta_k)\|_{R_{k+1}^{-1}}^2 + \frac{\ell}{2}\|\varepsilon_k\|_{R_{k+1}^{-1}}^2.$$

Substituting these bounds into equation 54 yields

$$\begin{aligned}\mathcal{L}(\theta_{k+1}) \leq &\mathcal{L}(\theta_k) - \frac{\eta}{2}\|\nabla\mathcal{L}(\theta_k)\|_{R_{k+1}^{-1}}^2 + \frac{\eta-\ell}{2}\|r_{k+1}\|_{R_{k+1}^{-1}}^2 \\ &+ \frac{\ell}{2}\|\varepsilon_k\|_{R_{k+1}^{-1}}^2 + \frac{L}{2}\|\theta_{k+1}-\theta_k\|^2.\end{aligned} \tag{55}$$

By Assumption 3 and equation 43, we have

$$\|\nabla\mathcal{L}(\theta_k)\|^2_{R_{k+1}^{-1}} \geq \nu_{\min}\|\nabla\mathcal{L}(\theta_k)\|^2,$$

$$\|r_{k+1}\|^2_{R_{k+1}^{-1}} \leq \nu_{\max}\|r_{k+1}\|^2,$$

and

$$\mathbb{E}\|\varepsilon_k\|^2_{R_{k+1}^{-1}} \leq \nu_{\max}\mathbb{E}\|\varepsilon_k\|^2 \leq \nu_{\max}\frac{\sigma^2}{|\mathcal{B}_k|}.$$

Taking expectation in equation 55, we obtain

$$\begin{aligned}\mathbb{E}\mathcal{L}(\theta_{k+1}) - \mathbb{E}\mathcal{L}(\theta_k) \leq &-\frac{\eta\nu_{\min}}{2}\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 + \frac{(\eta-\ell)\nu_{\max}}{2}\mathbb{E}\|r_{k+1}\|^2 \\ &+ \frac{L}{2}\mathbb{E}\|\theta_{k+1}-\theta_k\|^2 + \frac{\ell\nu_{\max}}{2}\frac{\sigma^2}{|\mathcal{B}_k|}.\end{aligned} \tag{56}$$

Combining equation 56 with the residual estimate equation 49, we get

$$\begin{aligned}\mathbb{E}V_{k+1} - \mathbb{E}V_k \leq &-\frac{\eta\nu_{\min}}{2}\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 \\ &- \left[c_1C_r - \frac{(\eta-\ell)\nu_{\max}}{2}\right]\mathbb{E}\|r_{k+1}\|^2 \\ &- (c_2 - c_1C_\theta)\mathbb{E}\|\theta_k-\theta_{k-1}\|^2 + \left(c_2+\frac{L}{2}\right)\mathbb{E}\|\theta_{k+1}-\theta_k\|^2 \\ &+ \left(c_1C_\varepsilon + \frac{\ell\nu_{\max}}{2}\right)\frac{\sigma^2}{|\mathcal{B}_k|} + c_1C_\varepsilon\frac{\sigma^2}{|\mathcal{B}_{k-1}|},\end{aligned} \tag{57}$$

Next, substituting equation 52 into equation 57 yields

$$\begin{aligned}\mathbb{E}V_{k+1} - \mathbb{E}V_k \leq &-d_1\mathbb{E}\|r_{k+1}\|^2 - d_2\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 \\ &+ (c_1C_\theta - c_2)\mathbb{E}\|\theta_k-\theta_{k-1}\|^2 + C_{v,+}\frac{\sigma^2}{|\mathcal{B}_k|} + C_{v,-}\frac{\sigma^2}{|\mathcal{B}_{k-1}|},\end{aligned} \tag{58}$$

where

$$d_1 := c_1C_r - \frac{(\eta-\ell)\nu_{\max}}{2} - 3\left(c_2+\frac{L}{2}\right)(\eta-\ell)^2\nu_{\max}^2, \tag{59}$$

$$d_2 := \frac{\eta\nu_{\min}}{2} - 3\left(c_2+\frac{L}{2}\right)\eta^2\nu_{\max}^2, \tag{60}$$

and

$$C_{v,+} := c_1C_\varepsilon + \frac{\ell\nu_{\max}}{2} + 3\left(c_2+\frac{L}{2}\right)\ell^2\nu_{\max}^2, \qquad C_{v,-} := c_1C_\varepsilon. \tag{61}$$

Choose $c_1, c_2 > 0$ and the algorithmic parameters in a stable regime such that

$$c_1C_\theta - c_2 \leq 0, \qquad d_1 > 0, \qquad d_2 > 0, \qquad \eta \geq \ell.$$

Then the nonpositive term $(c_1C_\theta - c_2)\mathbb{E}\|\theta_k-\theta_{k-1}\|^2$ can be dropped from the upper bound. Define

$$C_v := \max\{C_{v,+}, C_{v,-}\} > 0.$$

We finally obtain

$$\mathbb{E}V_{k+1} - \mathbb{E}V_k \leq -d_1\mathbb{E}\|r_{k+1}\|^2 - d_2\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 + C_v\sigma^2\left(\frac{1}{|\mathcal{B}_k|}+\frac{1}{|\mathcal{B}_{k-1}|}\right). \tag{62}$$

This proves Lemma 1. □

Table 3: Initial learning rates used in the learning-rate search.

Task	RADAR	RAD	Adam	AdamW	NAdam	Adan	AdaBelief	Lion
CIFAR-10, ViT (6.3M)	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	2.5×10^{-3}	1×10^{-3}	1×10^{-4}
CIFAR-100, ResNet-50	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	2.5×10^{-3}	1×10^{-3}	1×10^{-4}
Wikitext-103, GPT-2 Small	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	7.5×10^{-4}	3×10^{-4}	3×10^{-5}
FineWeb-Edu-10B, Qwen2.5-0.5B	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	7.5×10^{-4}	3×10^{-4}	3×10^{-5}
WikiText-2 SFT, GPT-2 Small	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	2.5×10^{-4}	1×10^{-4}	1×10^{-5}

A.5.3 Proof of Theorem 2

Proof. From equation 62, we have

$$d_1\mathbb{E}\|r_{k+1}\|^2 + d_2\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 \leq \mathbb{E}V_k - \mathbb{E}V_{k+1} + C_v\sigma^2\left(\frac{1}{|\mathcal{B}_k|} + \frac{1}{|\mathcal{B}_{k-1}|}\right). \tag{63}$$

Since $d_1\mathbb{E}\|r_{k+1}\|^2 \geq 0$, dropping it gives

$$d_2\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 \leq \mathbb{E}V_k - \mathbb{E}V_{k+1} + C_v\sigma^2\left(\frac{1}{|\mathcal{B}_k|} + \frac{1}{|\mathcal{B}_{k-1}|}\right). \tag{64}$$

Summing equation 64 from $k = 1$ to $T - 1$ gives

$$d_2\sum_{k=1}^{T-1}\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 \leq \mathbb{E}V_1 - \mathbb{E}V_T + C_v\sigma^2\sum_{k=1}^{T-1}\left(\frac{1}{|\mathcal{B}_k|} + \frac{1}{|\mathcal{B}_{k-1}|}\right). \tag{65}$$

Since $V_T \geq \mathcal{L}(\theta_T) \geq \mathcal{L}_*$, we have

$$\mathbb{E}V_1 - \mathbb{E}V_T \leq \mathbb{E}V_1 - \mathcal{L}_*.$$

Therefore,

$$d_2\sum_{k=1}^{T-1}\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 \leq \mathbb{E}V_1 - \mathcal{L}_* + C_v\sigma^2\sum_{k=1}^{T-1}\left(\frac{1}{|\mathcal{B}_k|} + \frac{1}{|\mathcal{B}_{k-1}|}\right). \tag{66}$$

Dividing both sides by $d_2(T-1)$ yields

$$\frac{1}{T-1}\sum_{k=1}^{T-1}\mathbb{E}\|\nabla\mathcal{L}(\theta_k)\|^2 \leq \frac{\mathbb{E}V_1 - \mathcal{L}_*}{d_2(T-1)} + \frac{C_v\sigma^2}{d_2(T-1)}\sum_{k=1}^{T-1}\left(\frac{1}{|\mathcal{B}_k|} + \frac{1}{|\mathcal{B}_{k-1}|}\right), \tag{67}$$

which proves Theorem 2. □

B Additional Details and Results on Numerical Experiments

B.1 Learning-Rate Search Protocol

Following the optimizer-specific recommendations, the initial learning rate of Adan was set to 2.5 times the task-specific base learning rate, while that of Lion was set to 0.1 times the base learning rate. All other optimizers used the task-specific base learning rate directly.

For each optimizer with an initial task-specific learning rate η_0, we evaluated the following candidate set:

$$\{0.1\eta_0,\ 0.5\eta_0,\ \eta_0,\ 5\eta_0,\ 10\eta_0\}. \tag{68}$$

The initial learning rates are summarized in Table 3.

Learning-rate tuning was performed using an independent tuning seed ($s = 5$). Each optimizer was then trained again from scratch using the selected learning rate and five independent final seeds:

$$\{0, 1, 2, 3, 4\}. \tag{69}$$

For all supervised learning tasks, the learning rate was selected using the mean of the final three validation losses to reduce sensitivity to a single evaluation point. For GPT-2 pre-training, each

Table 4: Selected learning rates across tasks and optimizers.

Task	RADAR	RAD	Adam	AdamW	NAdam	Adan	AdaBelief	Lion
CIFAR-10, ViT (6.3M)	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}	1.25×10^{-3}	5×10^{-4}	1×10^{-4}
CIFAR-100, ResNet-50	1×10^{-3}	5×10^{-3}	1×10^{-3}	5×10^{-3}	1×10^{-3}	2.5×10^{-3}	5×10^{-3}	5×10^{-4}
WikiText-103, GPT-2 Small	1.5×10^{-3}	3×10^{-4}	3×10^{-4}	3×10^{-4}	1.5×10^{-3}	3.75×10^{-3}	3×10^{-4}	1.5×10^{-4}
FineWeb-Edu-10B, Qwen2.5-0.5B	1.5×10^{-3}	1.5×10^{-3}	1.5×10^{-3}	1.5×10^{-3}	1.5×10^{-3}	3.75×10^{-3}	3×10^{-4}	1.5×10^{-4}
WikiText-2 SFT, GPT-2 Small	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	2.5×10^{-4}	1×10^{-4}	1×10^{-4}
HalfCheetah-v4, SAC	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	2.5×10^{-3}	1×10^{-3}	1×10^{-4}
Walker2d-v4, SAC	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	2.5×10^{-3}	1×10^{-3}	1×10^{-4}

Table 5: RADAR hyperparameters used in the reported experiments.

Setting	η	β_1	β_2	γ	ℓ
CIFAR-10, ViT (6.3M)	5e−4	0.9	0.999	0.1	5e−6
CIFAR-100, ResNet-50	1e−3	0.9	0.999	0.1	1e−5
WikiText-103, GPT-2 Small	1.5e−3	0.9	0.999	0.1	1.5e−5
FineWeb-Edu-10B, Qwen2.5-0.5B	1.5e−3	0.9	0.95	0.1	1.5e−5
WikiText-2 SFT, GPT-2 Small	1e−4	0.9	0.999	0.1	1e−6
HalfCheetah-v4, SAC	1e−3	0.9	0.999	0.01	1e−5
Walker2d-v4, SAC	1e−3	0.9	0.999	0.01	1e−5

candidate was trained for 1,000 optimizer steps, with validation losses recorded every 100 steps; the values at steps 800, 900, and 1,000 were used for selection. For Qwen2.5-0.5B pre-training, each candidate was trained for approximately 0.1B tokens, one tenth of the full 1B-token budget. For the ViT and ResNet-50 tasks, each candidate was trained for 10 epochs, and the validation losses from the final three epochs were used. For GPT-2 fine-tuning, each candidate was trained for one complete epoch, and the final three validation loss measurements were used.

More generally, for a task-specific validation metric $\mathcal{M}_{\mathrm{val}}$, the selection score was defined as

$$S(\eta) = \frac{1}{3} \sum_{j=M-2}^{M} \mathcal{M}_{\mathrm{val}}^{(j)}(\eta), \tag{70}$$

where M is the total number of validation measurements and $\mathcal{M}_{\mathrm{val}}$ denotes validation loss. The candidate with the lowest selection score was selected.

Within those tasks, all learning-rate trials used the same tuning seed, model initialization, data split and data order, batch configuration, warmup setting, weight-decay setting, and the same initial portion of the learning-rate schedule used in the final training runs.

B.2 Hyperparameters of All Experiments

Table 4 summarizes the learning rates used in the reported experiments. Table 5 summarizes the RADAR-specific hyperparameters. We set $\beta_1 = 0.9$ for all optimizers except Adan, which uses $(\beta_1, \beta_2, \beta_3) = (0.98, 0.92, 0.99)$. For RADAR and RAD, we set $\delta = 1$ and $\zeta = 10^{-16}$ in all experiments. For the other methods with a second-moment state, the corresponding coefficient is set to 0.999 or 0.99, except in the Qwen setting, where it is set to 0.95. Lion has no second-moment state.

B.3 Language Modeling

We provide implementation details for the three language-modeling settings described in Section 5.1: GPT-2 pre-training on WikiText-103, Qwen2.5-0.5B pre-training on FineWeb-Edu, and GPT-2 fine-tuning on WikiText-2. The configurations are summarized in Table 6.

GPT2-Small pre-training. We pre-train a GPT-2 model from scratch on WikiText-103. The model is initialized from the GPT-2 configuration without loading pretrained weights, while the GPT-2 tokenizer is used for tokenization. Empty lines are removed from the raw corpus, and the remaining

Table 6: Hyperparameter settings for language modeling experiments.

Setting	WikiText-103 + GPT-2 Small	FineWeb-Edu-10B + Qwen2.5-0.5B	WikiText-2 + GPT-2 Small
Training type	Pre-training from scratch	Pre-training from scratch	Fine-tuning
Training budget	10,000 steps	1B tokens (1,908 steps)	3 epochs
Sequence length / block size	512	2048	512
Per-device batch size	16	4	16
Gradient accumulation	4	16	4
Effective batch size	64	256	64
LR scheduler	Cosine	Cosine	Cosine
Warmup	500 steps	40 steps	0 steps
Max gradient norm	1.0	1.0	1.0
Precision	FP16	BF16	FP16
Logging frequency	Every 10 steps	Every 10 steps	Every step
Evaluation frequency	Every 500 steps	Every 100 steps	Every 10 steps

Table 7: Wall-clock time and peak GPU memory usage on GPT-2 pre-training. Results are averaged over five random seeds.

Optimizer	Total Wall Time (s)	Peak CUDA Allocated (GiB)
RADAR	4450.2022	12.8053
AdamW	4496.5346	12.8053
NAdam	4500.5386	12.8053

text is tokenized, concatenated, and split into fixed-length blocks of 512 tokens. The model is trained for 10,000 update steps with an effective batch size of 64, FP16 mixed precision, a cosine learning-rate scheduler with 500 warmup steps, and gradient clipping with maximum norm 1.0.

Qwen2.5-0.5B pre-training. We further evaluate large-scale language-model pre-training by training Qwen2.5-0.5B from scratch on FineWeb-Edu. The model is initialized from the Qwen2.5-0.5B configuration without loading pretrained weights, while the corresponding tokenizer is used for tokenization. The script shuffles the locally available FineWeb-Edu parquet records with seed 42, selects the first 2,000,000 documents for training and the next 10,000 for validation, then removes empty texts before tokenization. Each split is concatenated and divided into blocks of 2,048 tokens. This is a held-out validation split; the script does not define a separate test split. We train the model for a total budget of 1B tokens on 4$\times$ NVIDIA A100 GPUs, with a per-device batch size of 4 and 16 gradient-accumulation steps, resulting in an effective batch size of 256. Training uses BF16 mixed precision, a cosine learning-rate scheduler with 40 warmup steps, and gradient clipping with a maximum norm of 1.0. Validation loss is evaluated every 100 update steps.

GPT2-Small fine-tuning. We fine-tune a pretrained GPT-2 model on the WikiText-2 raw dataset. The input texts are tokenized using the GPT-2 tokenizer, concatenated, and split into blocks of 512 tokens. The model is trained for 3 epochs with an effective batch size of 64, FP16 mixed precision, a cosine learning-rate scheduler without warmup, and gradient clipping with maximum norm 1.0. Validation loss is evaluated every 10 steps.

B.4 Wall-Clock Time and GPU Memory Overhead on GPT-2 Pre-Training

A direct implementation of RADAR may appear to require an additional gradient state because its corrected momentum contains the gradient difference. Specifically, the momentum recursion can be written as

$$M_k = \beta_1 M_{k-1} + (1 - \beta_1 + \gamma)G_k - \gamma G_{k-1}. \tag{71}$$

Naively evaluating this recursion requires storing the previous gradient G_{k-1} in addition to the first- and second-moment states.

However, this additional state can be eliminated by an algebraic reparameterization. Let

$$\bar{M}_k = \beta_1 \bar{M}_{k-1} + (1 - \beta_1)G_k \tag{72}$$

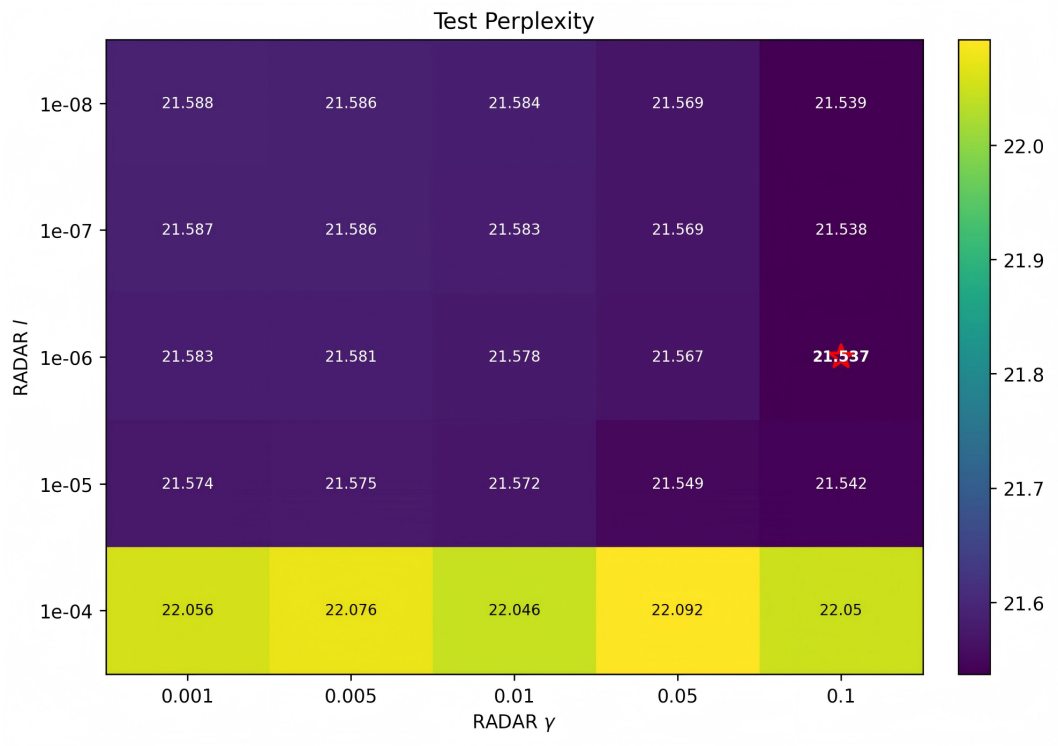

Figure 2: Test perplexity (PPL) for different ℓ and γ values on WikiText-2 GPT-2 fine-tuning; lower is better.

denote the standard exponential moving average of the gradients. Then the corrected momentum admits the equivalent representation

$$M_k = (\beta_1 - \gamma)\bar{M}_{k-1} + (1 - \beta_1 + \gamma)G_k. \tag{73}$$

Therefore, RADAR only needs to maintain the standard first-moment state $\bar{M}_k$ and the second-moment state, without explicitly storing G_{k-1}. Its persistent optimizer-state memory is thus the same order as that of AdamW.

As shown in Table 7, the optimized RADAR implementation has essentially the same computational cost as AdamW. RADAR requires 4450.20 seconds compared with 4496.53 seconds for AdamW, while both methods reach the same peak CUDA-allocated memory of 12.8053 GiB. NAdam exhibits a similar runtime of 4500.54 seconds. These results indicate that the residual correction in RADAR can be implemented without measurable additional GPU-memory overhead and with wall-clock time comparable to standard adaptive optimizers.

B.5 Sensitivity

We examine the sensitivity of RADAR to ℓ and γ using a 5×5 grid on WikiText-2 GPT-2 fine-tuning. We also vary δ and ζ separately around their default values.

Figure 2 shows that, for $\ell \in [10^{-8}, 10^{-5}]$, test perplexity changes modestly across the plotted γ values. The lowest displayed value is approximately 21.537, obtained at $\ell = 10^{-6}$ and $\gamma = 0.1$, which is the default configuration for this task. When ℓ increases to 10^{-4}, test perplexity rises to approximately 22.05–22.09 across the plotted γ values.

Varying δ by $\pm 10\%$ around 1 changes test perplexity by at most 0.0595 in the corresponding experiments. Varying ζ from 10^{-18} to 10^{-14} changes it by less than 0.01. These results indicate limited sensitivity to δ and ζ over the tested ranges.

The sensitivity experiments support the selected hyperparameters for WikiText-2 fine-tuning. In the five-seed main experiment, the hyperparameters were fixed across seeds.