ACCELERATING LLM PRETRAINING WITH ZEROTH-ORDER CURVATURE ON IDLE SERVING COMPUTE

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM pretraining is increasingly compute-intensive, while substantial inference serving compute remains idle during off-peak periods. However, such serving hardware is difficult to use for conventional first-order training due to its memory, networking and other system requirements. Zeroth-order (ZO) optimization offers a natural interface to this forward-only compute resource, but its gradient estimates break down in pretraining. Unlike fine-tuning, which can confine updates to a small parameter subspace, pretraining must update every parameter, and the alignment of a ZO gradient estimate with the true gradient shrinks as the number of updated parameters grows. We make a different observation: noisy ZO estimates can still provide useful curvature information, even when they are too inaccurate to serve as gradients. Based on this insight, we propose ZO-Curv, which leverages idle forward-only devices to estimate curvature and uses it to adapt and refine the update used during first-order training, thereby accelerating the process. We provide convergence guarantees that characterize the effect of finite-query curvature estimation. Experiments on LLMs from 125M to 1.5B parameters show that ZO-Curv consistently accelerates pretraining. At 770M parameters, ZO-Curv reaches AdamW's 20K-step validation loss in 3.9 hours instead of 7.5 hours, while achieving a lower final loss of 2.70 versus 2.84 under the same step budget. These results show that idle model serving compute can accelerate LLM pretraining when zeroth-order queries are used to estimate curvature rather than gradients directly.

1 INTRODUCTION

LLM pretraining demands substantial compute, with model development requiring repeated runs over architectures, training schedules, data mixtures, and optimizer configurations (Khandelwal et al., 2024; Kaplan et al., 2020; Hoffmann et al., 2022; Zhang et al., 2022; Yang et al., 2021). Meanwhile, rapidly growing model serving compute, from inference clusters to local devices (Cai, 2025), represents a potentially valuable neglected resource for accelerating pretraining.

Serving compute is abundant but difficult to train on. Much of this serving capacity is underutilized. As shown in Figure 1(a), serving clusters are provisioned for peak demand, leaving substantial capacity idle during off-peak periods (Patel et al., 2024; Griggs et al., 2024; Li et al., 2023; Liu et al., 2026a). Across tens of thousands of production Kubernetes clusters, Cast AI reports only 5% average GPU utilization from January 2025 to April 2026 (Cast AI, 2026). However, this idle compute cannot readily join conventional first-order (FO) training. Training must additionally store activations for backpropagation, gradients, and optimizer states, resulting in a 3.5–$9\times$ memory overhead over inference (Figure 3(a)). It also requires optimized backward kernels, ultra high-speed networks and runtime support that serving platforms often lack (Kwon et al., 2023; Zhong et al., 2024).

Zeroth-order optimization: promise and challenge. Zeroth-order (ZO) optimization needs only forward passes, estimating gradients from loss differences under random perturbations (Spall, 1992; Nesterov & Spokoiny, 2017). This makes it a natural fit for forward-only hardware, and it has recently proven practical for *fine-tuning* LLMs at inference-level memory cost (Malladi et al., 2023; Zhang et al., 2024). Pretraining is harder: the variance of random-direction ZO estimators grows with the parameter dimension d (Nesterov & Spokoiny, 2017; Duchi et al., 2015; Shamir, 2017). At the hundreds of millions to billions of parameters of modern LLMs, this leaves low-query estimates

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.

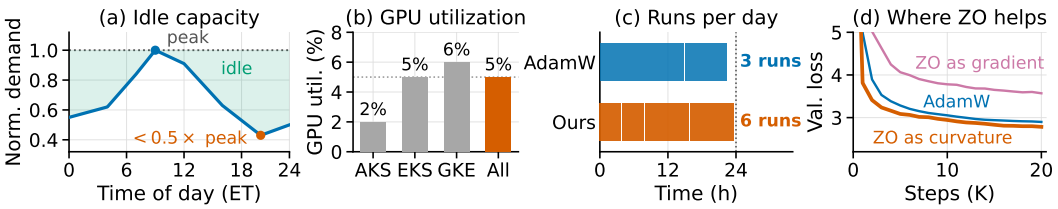

Figure 1: Landscape of idle serving devices and the pathway of harvesting them. (a) Diurnal inference demand leaves peak-provisioned capacity idle. (b) Production clusters exhibit low average GPU utilization. (c) Harvesting idle devices doubles the pretraining runs a cluster completes per day. (d) Comparison of ZO curvature and ZO gradients on pretraining.

extremely noisy. MeZO-style pretraining of GPT-2 355M confirms the effect, stalling near a validation loss of 8.5 while AdamW drops below 3.0 after the same number of training steps. Mixing ZO estimates into the FO gradient inherits the same noise and underperforms AdamW (Figure 1(d)).

Our approach: ZO-Curv. ZO-Curv uses ZO to estimate *curvature* rather than gradients. Unlike a gradient, curvature need not identify a precise descent direction, which noisy ZO queries cannot provide at pretraining scale; it only sets a coordinate-wise scale for the update, and even a coarse diagonal estimate can tell how aggressively each coordinate should move. ZO-Curv therefore keeps exact FO gradient computation on the training node and outsources only curvature estimation to idle serving devices, which evaluate perturbed losses with forward passes. The training node aggregates these estimates into a diagonal preconditioner for its FO update.

Our contributions are as follows.

- We show that ZO estimates, while too noisy to provide reliable descent directions for LLM pretraining, remain effective for estimating curvature to precondition FO gradients.
- We propose ZO-Curv, a hybrid optimization framework that keeps exact FO gradient computation on the training node while offloading forward-only curvature estimation to idle serving devices. We further establish convergence guarantees for ZO-Curv.
- We demonstrate consistent gains across models from 125M to 1.5B parameters. ZO-Curv reaches AdamW's target loss up to $2.2\times$ faster, with larger models benefiting more from additional serving compute. On the downstream task LAMBADA, ZO-Curv improves accuracy by 26.9% over AdamW.

2 Background and Motivation

2.1 Preliminaries

LLM pretraining. Pretraining minimizes the autoregressive negative log-likelihood $F(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\big[-\sum_i \log p_\theta(x_i \mid x_{<i})\big]$ over a large corpus $\mathcal{D}$, where $\theta \in \mathbb{R}^d$ are the transformer parameters. The de facto optimizer is AdamW (Loshchilov & Hutter, 2017): every step runs a forward and a backward pass and maintains two moment buffers alongside parameters and gradients.

Second-order preconditioning. Sophia (Liu et al., 2024) accelerates pretraining by dividing the gradient EMA (Exponential Moving Average) by an EMA of a diagonal Hessian estimate, followed by coordinate-wise clipping. The curvature estimate needs to be neither exact nor fresh: it is refreshed only every ten steps, and the EMA, the clipping, and a positive floor on the denominator absorb estimation noise. This coarse signal halves the number of steps to reach a given validation loss. Its cost, however, is charged to the training node, since the Gauss–Newton–Bartlett (GNB) estimator computes a minibatch gradient with resampled labels, an additional forward and backward pass. Sophia accounts for this as 5% extra compute per step on average, or roughly 50% extra FLOPs on each refresh step. Measured on GPU, however, a refresh step costs about twice an AdamW step rather than the $1.5\times$ that its FLOPs suggest (Section 2.3).

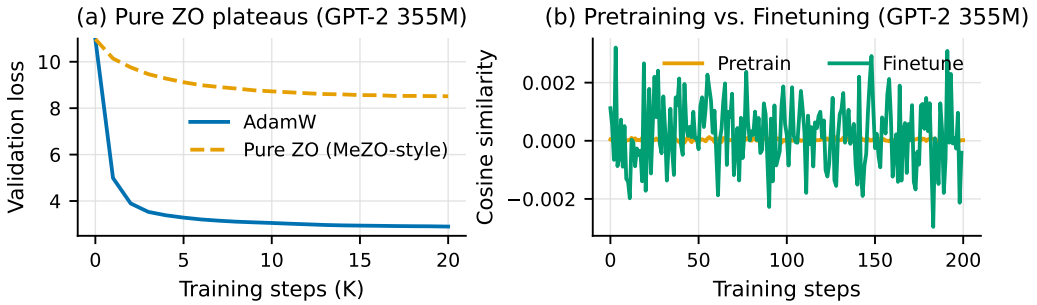

Figure 2: ZO gradients fail at pretraining. (a) Pure ZO stalls early while AdamW converges under the same budget. (b) Cosine similarity between the ZO gradient and the exact gradient.

Zeroth-order optimization. Zeroth-order (ZO) methods optimize with forward evaluations only (Spall, 1992; Nesterov & Spokoiny, 2017). The two-point estimator perturbs the parameters along a random direction u and forms

$$\hat{g} = \frac{F(\theta + \varepsilon u) - F(\theta - \varepsilon u)}{2\varepsilon} u, \tag{1}$$

which is unbiased for the gradient of a smoothed loss. Its variance, however, grows linearly with the dimension d at a fixed number of evaluations (Nesterov & Spokoiny, 2017; Duchi et al., 2015). Because each evaluation needs only inference-level memory, ZO has recently proven practical for *fine-tuning* LLMs (Malladi et al., 2023; Zhang et al., 2024), which raises the question of whether it can support pretraining as well.

2.2 ZO GRADIENTS LOSE DIRECTIONAL SIGNAL AT PRETRAINING SCALE

The most direct way to use forward-only serving devices is to let ZO replace backpropagation outright. Figure 2(a) shows that it does not work. We pretrain a 355M model in GPT-2 structure from scratch on OpenWebText with a MeZO-style two-point estimator, against AdamW under an identical setup. AdamW drops below a validation loss of 3.0 within the budget, whereas pure ZO stalls near 8.5 after an initial descent. The gap is not a constant-factor slowdown: beyond the first few thousand steps, the ZO run makes essentially no further progress.

The failure is directional and scales with the number of trainable parameters d. Figure 2(b) compares the cosine similarity between the two-point estimate equation 1 and the exact gradient in pretraining ($d \approx 3.5 \times 10^8$) and in LoRA fine-tuning of the same model on SST-2 (Socher et al., 2013) ($d \approx 7.9 \times 10^5$). Because the estimate is parallel to its Gaussian direction u, this cosine has expected magnitude close to $\sqrt{2/(\pi d)}$ for any gradient, matching both curves: it reflects the dimension, not the loss landscape. Fine-tuning can confine the search to a small subspace, but pretraining must update every parameter, so each ZO estimate carries almost no directional information. Forward-only devices should therefore supply not the descent direction but information that tolerates far noisier measurements.

2.3 SERVING DEVICES CANNOT TRAIN, AND CURVATURE IS COSTLY

Could idle serving devices instead join conventional training as ordinary data-parallel workers? Figure 3(a) compares peak per-device memory for forward-only execution against AdamW and SophiaG training across model sizes. Forward-only execution fits a representative 24 GB serving card at every size, peaking at 8.5 GB for the 1.5B model. Training crosses that limit already at 355M and reaches roughly eight times the forward-only footprint at 1.5B, because it additionally holds activations for backpropagation, gradients, and optimizer state. A device that comfortably serves a model still cannot train it.

Second-order preconditioning strains the training node further, more than its FLOPs accounting indicates. Section 2.1 reports Sophia consumes roughly 50% extra FLOPs for the GNB refresh. While measured on GPU (Figure 3(b)), a step that recomputes the GNB estimate takes 2.12× the

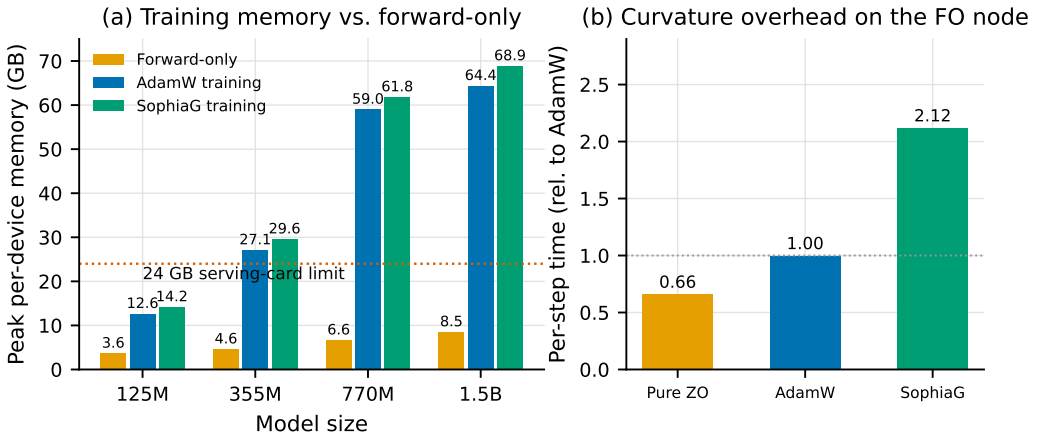

Figure 3: Serving devices cannot join training, and curvature estimation burdens the training node. (a) Forward-only execution fits a 24 GB serving card at every size; AdamW and SophiaG training exceed it from 355M onward. (b) A SophiaG curvature-refresh step costs about twice an AdamW step, while a forward-only computation is much cheaper to obtain.

time of an AdamW step. Meanwhile, forward-only ZO estimation costs only 66%, and it runs within serving-level memory. The compute that curvature estimation needs is cheap and abundant, but it sits on the wrong side of the cluster.

Together, the two negative results motivate a single split. The first-order gradient is indispensable (Section 2.2) and demands full training capability, so it must stay on the training node. Curvature is auxiliary: it only rescales coordinates, the update rule that consumes it is built to tolerate noise (EMA, clipping, positive floor), and is friendly to ZO estimation on the idle serving nodes.

3 ZO-Curv: Curvature from Idle Forward-Only Compute

3.1 Setting

Figure 4 illustrates the overall ZO-Curv workflow. We consider a service cloud with one strong training node and M forward-only serving nodes. The training node supports backpropagation and maintains optimizer states; serving nodes hold the model and evaluate losses with inference-level memory. The nodes are physically interconnected within the same cloud, and we assume that transferring model snapshots and query results is not the dominant cost.

Let $F(\theta) = \mathbb{E}_{x\sim\mathcal{D}}[\ell(\theta; x)]$ be the pretraining objective, with $\theta \in \mathbb{R}^d$. The two pools run concurrently, offset by one step. At step t, the training node broadcasts the snapshot θ_t to the serving pool and computes the exact gradient $g_t = \nabla L_t(\theta_t)$ with true next-token labels; meanwhile, the pool evaluates its curvature queries at θ_t, whose results arrive in time for step $t+1$. The update at step t therefore uses the curvature estimated at θ_{t-1}: neither side ever waits for the other, and the curvature EMA and the model are still refreshed at every training step. The one-step offset has no measurable effect on training (Appendix F).

3.2 Forward-only diagonal curvature estimation

Estimation target. Our target is the Gauss–Newton–Bartlett (GNB) diagonal estimator used by Sophia-G (Liu et al., 2024). Given a curvature minibatch $\mathcal{B}_t = \{x_b\}_{b=1}^{B}$, we sample pseudo-labels $\hat{y}_b \sim p_{\theta_t}(\cdot \mid x_b)$ and define the sampled-label loss $\ell_t^{\text{samp}}(\theta) = B^{-1}\sum_{b=1}^{B} \ell_{\text{ce}}(f_\theta(x_b), \hat{y}_b)$. Writing $c_t = \nabla \ell_t^{\text{samp}}(\theta_t)$, the reference estimate is $h_t^{\text{G}} = B\, c_t \odot c_t$. Its expectation over the sampled labels recovers the minibatch Gauss–Newton diagonal (Martens, 2020). We estimate a damped version of this GNB diagonal from forward passes only, avoiding the additional backward pass needed to compute c_t.

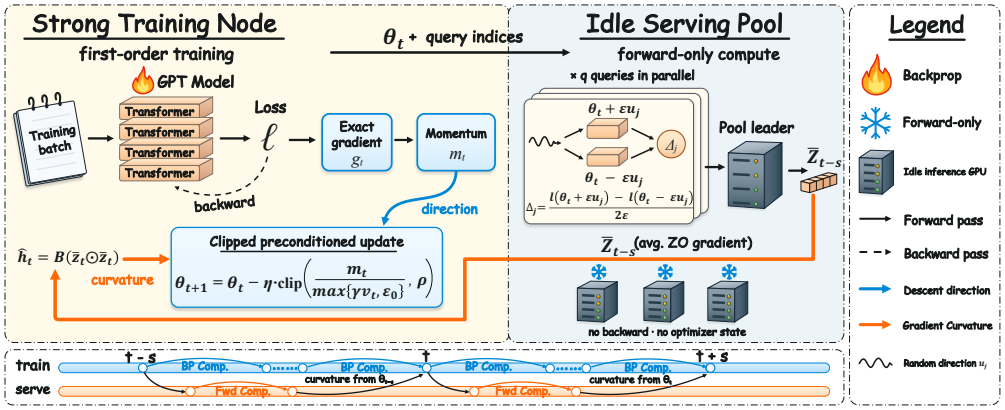

Figure 4: **ZO-Curv overview.** The strong node runs ordinary first-order pretraining; idle serving devices concurrently evaluate q central-difference queries and return their averaged directional estimate, from which the strong node assembles the diagonal curvature that preconditions its update.

Forward-only queries. At each step, the serving pool draws q independent full-parameter directions $u_{t,j} \sim \mathcal{N}(0, I_d)$ and evaluates two perturbed losses per direction. All queries share the same snapshot, curvature minibatch, and sampled labels, which are held fixed under perturbation. The returned central differences determine the curvature estimate:

$$\Delta_{t,j} = \frac{\ell_t^{\text{samp}}(\theta_t + \varepsilon u_{t,j}) - \ell_t^{\text{samp}}(\theta_t - \varepsilon u_{t,j})}{2\varepsilon},$$
$$\bar{z}_t = \frac{1}{q}\sum_{j=1}^{q} \Delta_{t,j} u_{t,j}, \qquad h_t^{\text{ZO}} = B\,\bar{z}_t \odot \bar{z}_t. \tag{2}$$

Averaging the directional estimates before squaring yields an unbiased estimate of the GNB diagonal with uniform damping $\|h_t^{\text{G}}\|_1/q$ (Proposition 1). The curvature minibatch and pseudo-labels are sampled on the serving nodes. Thus, the downlink carries only the model snapshot and query indices, while the pool aggregates the directional terms internally. A pool leader returns the averaged vector $\bar{z}_t$, requiring one gradient-sized transfer per refresh. Overall, the estimator requires $2q$ perturbed forward evaluations across the pool.

Preconditioned update. The training node keeps two exponential moving averages, both initialized at zero: the gradient momentum $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ and the curvature $v_t = \beta_2 v_{t-1} + (1-\beta_2) h_{t-1}^{\text{ZO}}$; the latter uses the estimate of the previous step, which the pool returns one step after receiving the snapshot (Section 3.1). Dividing the momentum by the floored curvature $\hat{d}_t = \max\{\gamma v_t, \epsilon_0\}$ and clipping the result yields

$$\theta_{t+1} = (1 - \eta_t \lambda)\theta_t - \eta_t \,\text{clip}(m_t/\hat{d}_t, \rho), \tag{3}$$

where γ scales the curvature, λ is the weight-decay coefficient, and all operations act coordinate-wise. The floor ϵ_0 and the clipping threshold ρ are what make this update forgiving of curvature error: a positive denominator rescales a coordinate without flipping its sign, and clipping caps the step wherever the estimated curvature is too small. Algorithms 1 and 2 state both procedures for a general staleness s, partitioning the q query indices into disjoint sets $\{\mathcal{J}_m\}_{m=1}^{M}$ across the pool.

3.3 Convergence Analysis

We study a deterministic core of equation 3: replace minibatch gradients with $g_t = \nabla F(\theta_t)$ and omit momentum and weight decay; the results hold for staleness s (Remark 2). Proofs and explicit constants are in Appendix H.

Proposition 1 (Forward-only damped GNB curvature)**.** Fix a step and condition on its curvature minibatch and sampled labels. As $\varepsilon \to 0$,

$$\mathbb{E}[h_t^{\text{ZO}}] = \tilde{h}_t := \Big(1 + \frac{1}{q}\Big) h_t^{\text{G}} + \lambda_t \mathbf{1}, \qquad \lambda_t = \frac{\|h_t^{\text{G}}\|_1}{q}, \tag{4}$$

Algorithm 1 ZO-Curv: training node

Input: initial parameters θ_1; learning rates $\{\eta_t\}_{t=1}^{T}$; query budget q; EMA coefficients β_1, β_2; curvature scale γ; clipping threshold ρ; denominator floor ϵ_0; weight decay λ; staleness s
$m_0 \leftarrow 0$; $v_0 \leftarrow 0$; partition $\{1, \dots, q\}$ into $\{\mathcal{J}_m\}_{m=1}^{M}$
Broadcast $(\theta_1, \{\mathcal{J}_m\})$; $t_{\mathrm{disp}} \leftarrow 1$
for $t = 1, \dots, T$ **do**
  $g_t \leftarrow \nabla L_t(\theta_t)$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  if $t - t_{\mathrm{disp}} = s$ **then**
    $h \leftarrow B\, \bar{z}_{t-s}^{\odot 2}$ ▷ queries of θ_{t-s}
    $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) h$
    Broadcast $(\theta_t, \{\mathcal{J}_m\})$; $t_{\mathrm{disp}} \leftarrow t$ ▷ non-blocking; pool runs Alg. 2
  else
    $v_t \leftarrow v_{t-1}$
  end if
  $\hat{d}_t \leftarrow \max\{\gamma v_t, \epsilon_0\}$
  $\theta_{t+1} \leftarrow (1 - \eta_t \lambda)\theta_t - \eta_t \,\mathrm{clip}(m_t / \hat{d}_t, \rho)$
end for

Algorithm 2 ZO-Curv: serving node m

Input: snapshot θ_t; query indices $\mathcal{J}_m$; finite-difference radius ε
Draw $\mathcal{B}$ and $\hat{y} \sim p_{\theta_t}$; form ℓ^{samp}
for $j \in \mathcal{J}_m$ **do**
  $u_j \sim \mathcal{N}(0, I_d)$
  $L^{\pm} \leftarrow \ell^{\mathrm{samp}}(\theta_t \pm \varepsilon u_j)$
  $\Delta_j \leftarrow (L^{+} - L^{-})/2\varepsilon$
end for
$z_m \leftarrow \sum_{j \in \mathcal{J}_m} \Delta_j u_j$ ▷ local partial sum
Send z_m to the pool leader
Leader: $\bar{z}_t \leftarrow q^{-1} \sum_{m=1}^{M} z_m$
Return $\bar{z}_t$ to the training node

and $\mathbb{E}[(h_{t,i}^{\mathrm{ZO}})^2] \le C_v \tilde{h}_{t,i}^2$ for a universal constant C_v.

The forward-only estimate is thus unbiased for a GNB diagonal with a Levenberg–Marquardt-type damping λ_t that decreases as $1/q$, and its fluctuation is of the order of its target in every coordinate, independently of d. Under Assumption A1, finite differences add a relative error of order ε^2.

Let $D_t = \mathrm{Diag}(d_t)$, $d_t = \max\{\gamma \tilde{v}_t, \epsilon_0\}$, be the denominator built from the EMA $\tilde{v}_t$ of $\tilde{h}_t$ at the same iterates, and let $\widehat{D}_t = \mathrm{Diag}(\hat{d}_t)$ be its ZO counterpart. Fresh directions at every refresh let the EMA average the fluctuations: if each coordinate of $\tilde{h}_t$ varies by at most a factor κ over the horizon, then with probability at least $1 - \zeta$, $(1-\delta)D_t \preceq \widehat{D}_t \preceq (1+\delta)D_t$ after $\mathcal{O}(1/(1-\beta_2))$ warm-up steps, where $\delta = \mathcal{O}\big(\kappa[\sqrt{(1-\beta_2)\Lambda} + (1-\beta_2)\Lambda^2] + \sqrt{\zeta/T} + \varepsilon^2\big)$ and $\Lambda = \log(qdT/\zeta)$ (Theorem 3); the positive floor preserves this relative bound. We require $\delta < 1$.

Theorem 1 (Robust descent)**.** Suppose $\nabla^2 F \preceq \alpha D_t$ along each update segment and the above distortion bound holds. For $\eta \le (1-\delta)/\alpha$, the deterministic update $\theta_{t+1} = \theta_t - \eta \,\mathrm{clip}(\widehat{D}_t^{-1} g_t, \rho)$ satisfies

$$F(\theta_{t+1}) \le F(\theta_t) - \frac{\eta}{2} \sum_i \min\left\{ \frac{g_{t,i}^2}{\hat{d}_{t,i}}, \rho |g_{t,i}| \right\}. \tag{5}$$

If F is bounded below, the minimum of this clipped stationarity measure over T steps is $\mathcal{O}(1/T)$.

A positive denominator preserves each gradient coordinate's sign, and clipping bounds the update magnitude; smoothness and the step-size condition ensure that the finite step decreases the loss.

Theorem 2 (Local linear convergence)**.** Suppose clipping is inactive and the preconditioned PL condition $g_t^{\top} D_t^{-1} g_t \ge 2\mu_D (F(\theta_t) - F_\star)$ holds in the local region. Under the conditions of Theorem 1, choosing $\eta = (1-\delta)/\alpha$ gives

$$F(\theta_{t+1}) - F_\star \le \left(1 - \frac{\mu_D}{\alpha} \frac{1-\delta}{1+\delta}\right) (F(\theta_t) - F_\star). \tag{6}$$

Curvature error enters only through the factor $(1-\delta)/(1+\delta)$, which is governed by the EMA window and depends on d only logarithmically. The budget q instead sets the damping of the reference preconditioner: as q grows, λ_t decreases and D_t approaches the undamped GNB preconditioner of Sophia-G. The serving pool thus supplies coordinate-wise update scales while the training node retains first-order gradient computation.

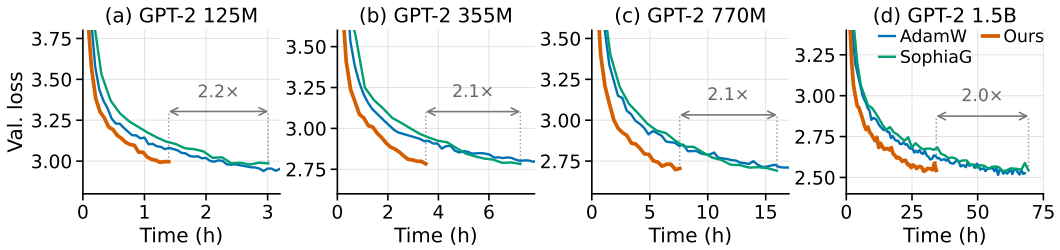

Figure 5: Main result (RQ1): forward-only curvature turns step efficiency into wall-clock speedup (OpenWebText; AdamW receives a doubled step budget). ZO-Curv reaches AdamW's final loss in roughly half the time at every size, whereas SophiaG spends its step advantage on curvature computed on the training node.

4 Experiments

We evaluate ZO-Curv to answer four research questions: **RQ1:** does forward-only curvature estimation accelerate LLM pretraining end to end, in wall-clock time and across model sizes? **RQ2:** how does the gain correlate with the query budget q? **RQ3:** can curvature queries substitute for first-order batch size, exchanging training-node compute for idle serving compute? **RQ4:** do faster pretraining loss curves translate into downstream quality?

4.1 Setup

Models and data. GPT-2–style decoder-only transformers (Radford et al., 2019) with 125M, 355M, and 770M parameters, plus a 1.5B GPT-NeoX configuration, trained from scratch on OpenWebText (Gokaslan & Cohen, 2019) with context length 1024 and global batch size 512. SophiaG and ZO-Curv train for 20K steps at 125M–770M and 50K steps at 1.5B; AdamW receives a doubled step budget (40K and 100K, respectively) so that all methods span a comparable wall-clock window.

Baselines. (i) AdamW (Loshchilov & Hutter, 2017), the FO standard. (ii) SophiaG (Liu et al., 2024), which computes the GNB curvature by backpropagation on the training node, refreshed at every step so that both second-order methods consume equally fresh curvature. (iii) Pure ZO, a MeZO-style ZO-SGD (Malladi et al., 2023). (iv) InstructedZO (Cai, 2025), which mixes ZO gradient estimates into the FO gradient and represents the gradient-level use of forward-only compute. (v) ZO-Curv, with $q = 512$ unless stated otherwise. Further details are given in Appendix B.

Hardware and timing. All experiments run on a single $8\times$H100 node. The forward-only pool is emulated on the same hardware: at every step, the $2q$ perturbed forward evaluations of Section 3.2 are computed alongside the first-order step. Wall-clock time for ZO-Curv counts the training-node critical path; the concurrent curvature evaluations are excluded, as they would occupy otherwise-idle serving devices. Section 4.2 additionally runs first-order training and the forward-only queries on disjoint GPU groups, and the speedup persists.

4.2 RQ1: End-to-end acceleration

Figure 5 plots validation loss against wall-clock time and grants AdamW twice the steps to span a comparable time window. Table 1 instead holds every method to the same 20K-step budget at 770M and measures the time to reach AdamW's final loss. ZO-Curv reaches this loss in roughly half the wall-clock time. The speedup is consistent from 125M to 1.5B.

Table 1 also shows that simply using forward-only compute does not provide this benefit. Pure ZO never reaches the target, while InstructedZO finishes above AdamW. Both use ZO estimates as descent directions. In contrast, ZO-Curv uses forward-only estimates for curvature while retaining the exact first-order gradient. The benefit therefore comes from using forward-only compute for curvature rather than for the descent direction.

The comparison with SophiaG further isolates *where* curvature is computed. SophiaG and ZO-Curv make nearly identical progress per step (Appendix C). However, SophiaG computes its GNB

Table 1: Main comparison at GPT-2 770M with a shared 20K-step budget. Steps- and time-to-target measure how quickly each method reaches AdamW's final validation loss; "–" = never reached.

Method	Val. loss ↓	Steps to target (K) ↓	Time to target (h) ↓
AdamW	2.84	20.0	7.53
SophiaG	2.69	10.0	7.98
Pure ZO (MeZO-style)	8.01	–	–
InstructedZO	3.37	–	–
ZO-Curv (ours)	2.70	10.5	**3.98**

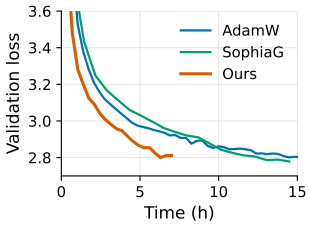

Figure 6: Real deployment on disjoint GPU groups. The x-axis reports the measured elapsed training time.

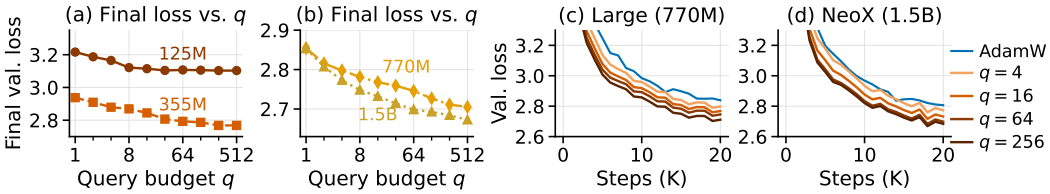

Figure 7: Effect of the query budget q across model sizes. (a)(b) Final validation loss for different query budgets; (c)(d) corresponding training trajectories.

curvature on the training node, where a refresh costs $2.12\times$ an AdamW step (Section 2.3). Table 1 shows the resulting wall-clock overhead. SophiaG reaches the target in only 10.0K steps but takes 7.98 hours, more than the 7.53 hours AdamW needs for its full run. ZO-Curv reaches the same target in 10.5K steps and takes only 3.98 hours. Its curvature estimation runs concurrently on the idle serving pool, leaving the training-node critical path close to that of AdamW. Refreshing curvature every ten steps, as in the original Sophia, amortizes the overhead only by using staler curvature. Even then, SophiaG remains 6–12% slower per step than ZO-Curv (Appendix D). Offloading curvature removes this training-node overhead rather than trading it against freshness.

Figure 6 further validates the speedup under a real concurrent deployment. We partition one $8\times$H100 node into two disjoint four-GPU groups. One group performs first-order training, while the other executes forward-only curvature queries. The returned curvature has a three-step staleness ($s = 3$). AdamW and SophiaG use only the four training GPUs, so the other four GPUs represent otherwise-idle serving capacity. Unlike the emulated setting above, the figure reports the measured elapsed training time of the concurrent runs. ZO-Curv completes 20K steps in about half the time required by SophiaG. At 20K steps, ZO-Curv reaches a validation loss of 2.81, close to SophiaG's 2.78, while requiring roughly half the elapsed training time. It also reaches this loss in less than half the time of AdamW. Thus, the wall-clock acceleration persists when curvature estimation and first-order training execute concurrently on disjoint GPU groups.

4.3 RQ2: Effect of the Query Budget

In Section 3.3, the damping of the curvature target decreases as $1/q$, bringing the preconditioner closer to the GNB diagonal of SophiaG, and Figure 7(a,b) confirms the effect on the final loss. The loss improves as q grows, monotonically until the curve flattens. Where it flattens depends on scale: 125M is essentially saturated by $q \approx 32$ and 355M by $q \approx 256$, while 770M and 1.5B still improve at $q = 512$. This ordering is consistent with the dimension dependence of the estimation error. Figure 7(c,d) shows the corresponding trajectories at the two largest sizes. Even $q = 4$ separates from AdamW within the first few thousand steps, and larger budgets lower the whole curve rather than only its endpoint. Appendix E repeats these panels for all four sizes.

The q queries are mutually independent, so the budget partitions freely across the pool and bounds how many idle devices can contribute in parallel. Read as a provisioning rule, the sweep says that larger models absorb more idle capacity before the returns flatten. Since a larger budget never costs accuracy, we use $q = 512$ in all other experiments.

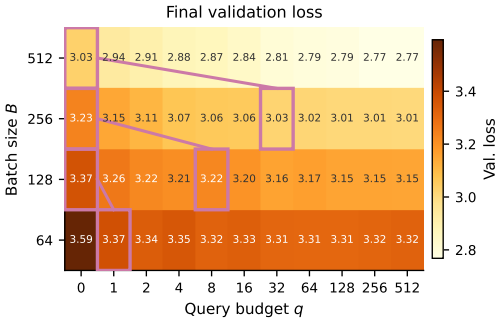

Figure 8: Effect of the training batch size B and query budget q on final validation loss for GPT-2 355M. The q=0 column corresponds to AdamW.

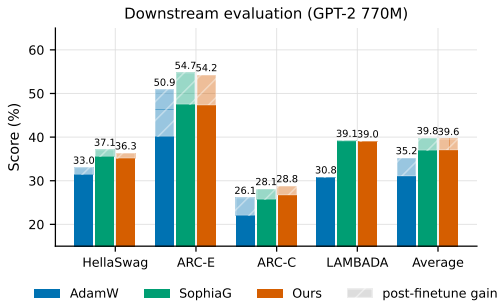

Figure 9: Downstream performance of different models. Solid bars: few-shot performance; hatched bars: LoRA finetuning gains.

4.4 RQ3: Trading first-order batch for queries

The experiments so far hold the first-order configuration fixed and add queries. We now ask the converse: can queries pay for a *smaller* first-order batch? Figure 8 varies the training batch size jointly with q at 355M, with the step budget held fixed across cells and the q=0 column giving plain AdamW. Along the marked pairs, halving the batch and adding a modest query budget recovers the full-batch loss: a batch of 256 with q=32 matches a batch of 512 at 3.03. Fewer queries are needed as the batch shrinks, eight when replacing a batch of 256 and a single one when replacing a batch of 128. A first-order step at half the batch costs roughly half the training-node compute, so this substitution shortens every step while the added queries run on serving hardware. ZO-Curv thus offers two ways to spend the same idle capacity: keep the batch and reach the target loss in less time (RQ1), or shrink the batch and cheapen every step at equal loss. Both trade scarce training GPUs for abundant serving compute.

4.5 RQ4: Downstream quality

We evaluate the 770M checkpoints at the 20K-step budget of Table 1 on HellaSwag (Zellers et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), and LAMBADA (Paperno et al., 2016). We first evaluate the pretrained checkpoints with few-shot prompting and then apply LoRA finetuning to each task (Figure 9; protocol in Appendix G). Before finetuning, ZO-Curv consistently outperforms AdamW across all four tasks and closely tracks SophiaG. The largest gain appears on LAMBADA, where ZO-Curv improves accuracy from 30.8% to 39.1%, a 26.9% relative improvement over AdamW.

The advantage persists after finetuning. ZO-Curv achieves an average score of 39.6, compared with 35.2 for AdamW and 39.8 for SophiaG. These results show that the lower pretraining loss of ZO-Curv translates into stronger downstream performance rather than remaining confined to the pretraining objective. Meanwhile, its near-parity with SophiaG shows that curvature estimated on forward-only hardware provides comparable downstream quality to curvature computed by backpropagation.

5 Conclusion

We show that idle serving compute can accelerate LLM pretraining by estimating curvature rather than gradients. ZO-Curv keeps exact first-order gradients on the training node while using noisy zeroth-order estimates to precondition step sizes. Across 125M–1.5B models, ZO-Curv matches SophiaG step for step and reaches AdamW's loss in roughly half the wall-clock time. Future work will evaluate ZO-Curv on larger real-world serving clusters.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, provide feedback on experimental design, interpret results, and assist with translation. We did not use generative AI tools to formulate the convergence results or to develop the core hypothesis or conceptual framework of ZO-Curv. These contributions originated with the authors. Generating synthetic datasets, cleaning or reformatting datasets, and conducting qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to search for, identify, and summarize related literature, to format references, and to support brainstorming. We have reviewed all AI-assisted work. AI-assisted implementations were reviewed and validated through the training runs reported in the paper. The authors verified the existence and bibliographic details of every cited work and revised all AI-drafted text. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3.2 and Algorithms 1 and 2 specify ZO-Curv completely, including the curvature estimator and the preconditioned update. Section 3.3 summarizes our theoretical results, and Appendix H states their assumptions and gives complete proofs. Section 4.1 describes the experimental protocol, and Appendix B lists the model configurations, all optimizer and estimator hyperparameters (Table 2), and the profiling protocol. Our implementation builds on the public nanoGPT code base used by Sophia (Liu et al., 2024), and all data are public: we pretrain on OpenWebText with the nanoGPT split and evaluate on standard benchmarks following the protocol of Appendix G. Appendix **??** describes the deployment on disjoint GPU groups. We will release our code upon acceptance.

REFERENCES

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

Atılım Güneş Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Maksim Riabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 558–568, 2023.

Dongqi Cai. Federated llm pre-training on mobile phones. In *Proceedings of the 23rd Annual International Conference on Mobile Systems, Applications and Services*, pp. 657–658, 2025.

Cast AI. 2026 state of kubernetes optimization report. `https://cast.ai/reports/kubernetes-optimization-report/`, 2026. Analysis of production Kubernetes clusters across AWS, GCP, and Azure.

Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Konstantinos Parasyris, Jiancheng Liu, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. Deepzero: Scaling up zeroth-order optimization for deep model training. In *International Conference on Learning Representations*, volume 2024, pp. 50185–50206, 2024.

Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order finetuning for language models with low-rank structures. In *International Conference on Learning Representations*, volume 2025, pp. 62581–62607, 2025.

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

Brannon Dorsey and Matt Kafonek. Borrowing the night: Reclaiming idle inference gpus for research. `https://runway.com/news/engineering/borrowing-the-night-reclaiming-idle-inference-gpus-for-research`, July 2026.

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

Aaron Gokaslan and Vanya Cohen. OpenWebText corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Tyler Griggs, Xiaoxuan Liu, Jiaxiang Yu, Doyoung Kim, Wei-Lin Chiang, Alvin Cheung, and Ion Stoica. M\'elange: Cost efficient large language model serving by exploiting gpu heterogeneity. *arXiv preprint arXiv:2404.14527*, 2024.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

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

Shuoran Jiang, Qingcai Chen, Youcheng Pan, Yang Xiang, Yukang Lin, Xiangping Wu, Chuanyi Liu, and Xiaobao Song. Zo-adamu optimizer: Adapting perturbation by the momentum and uncertainty in zeroth-order optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18363–18371, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Apoorv Khandelwal, Tian Yun, Nihal V Nayak, Jack Merullo, Stephen H Bach, Chen Sun, and Ellie Pavlick. $100K or 100 days: Trade-offs when pre-training with academic resources. *arXiv preprint arXiv:2410.23261*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Jiamin Li, Hong Xu, Yibo Zhu, Zherui Liu, Chuanxiong Guo, and Cong Wang. Lyra: Elastic scheduling for deep learning clusters. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pp. 835–850, 2023.

Fangyue Liu, Hua Liu, Xinyuan Lyu, Shuo Ai, Hao Liang, Lingpeng Chen, Ziqian Hu, Chong Zha, Xin Jin, Hanmei Luo, et al. Valve: Production online-offline inference colocation with jointly-bounded preemption latency and rate. *arXiv preprint arXiv:2604.07874*, 2026a.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International conference on learning representations*, volume 2024, pp. 1621–1650, 2024.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. Sparse mezo: Less parameters for better performance in zeroth-order llm fine-tuning. *Advances in Neural Information Processing Systems*, 38:21139–21169, 2026b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132. IEEE, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*, 2022.

Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pp. 29416–29440. PMLR, 2023.

Ohad Shamir. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *Journal of Machine Learning Research*, 18(52):1–11, 2017.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pp. 4596–4604. PMLR, 2018.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

James C Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE transactions on automatic control*, 45(10):1839–1853, 2000.

John Thorpe, Pengzhan Zhao, Jonathan Eyolfson, Yifan Qiao, Zhihao Jia, Minjia Zhang, Ravi Netravali, and Guoqing Harry Xu. Bamboo: Making preemptible instances resilient for affordable training of large {DNNs}. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 497–513, 2023.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 93423–93444, 2025.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

Ziming Yu, Pan Zhou, Sike Wang, Jia Li, Mi Tian, and Hua Huang. Zeroth-order fine-tuning of llms in random subspaces. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4475–4485. IEEE, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D Lee, Wotao Yin, Mingyi Hong, et al. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark. *arXiv preprint arXiv:2402.11592*, 2024.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Tsang. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. In *International Conference on Learning Representations*, volume 2025, pp. 43496–43520, 2025.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. {DistServe}: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, pp. 193–210, 2024.

A Related work

Zeroth-order optimization. ZO methods build gradient surrogates from function evaluations alone, from SPSA (Spall, 1992) to the smoothing analysis of Nesterov & Spokoiny (2017) and its stochastic nonconvex extension (Ghadimi & Lan, 2013). Their rates degrade with the dimension, and this dependence is optimal up to constants for two-point estimators (Duchi et al., 2015; Shamir, 2017). MeZO (Malladi et al., 2023) made ZO practical for LLM fine-tuning by regenerating perturbations from random seeds, and later work benchmarks ZO fine-tuning (Zhang et al., 2024) and reduces its variance with adaptive momentum (Jiang et al., 2024), sparse perturbations (Liu et al., 2026b), and low-rank subspaces (Chen et al., 2025; Yu et al., 2025). DeepZero (Chen et al., 2024) trains convolutional networks from scratch with coordinate-wise estimates and pruning-guided sparsity, and forward gradients (Baydin et al., 2022; Ren et al., 2022) replace backpropagation with forward-mode directional derivatives, whose variance likewise grows with the number of parameters. InstructedZO (Cai, 2025) mixes ZO estimates into first-order gradients to pretrain on mobile devices. All of these use the estimate as the descent direction; Section 2.2 shows that this fails at LLM pretraining scale, and ZO-Curv uses ZO queries only for curvature.

Zeroth-order curvature estimation. Second-order SPSA (Spall, 2000) estimates the Hessian from loss values alongside the gradient and uses it in a Newton-type update. HiZOO (Zhao et al., 2025) estimates a diagonal Hessian with one extra forward pass per step and uses it to precondition ZO fine-tuning of LLMs. Both still take the descent direction from ZO estimates, in settings that avoid backpropagation altogether. ZO-Curv instead pairs a forward-only curvature estimate with the exact first-order gradient, targets the GNB diagonal of Sophia-G for pretraining, and computes the estimate on separate idle hardware.

Preconditioned optimizers for pretraining. Adaptive methods (Kingma & Ba, 2014; Loshchilov & Hutter, 2017) precondition with gradient moments, Adafactor (Shazeer & Stern, 2018) factors these moments to save memory, and K-FAC (Martens & Grosse, 2015) and Shampoo (Gupta et al., 2018) use structured curvature, with the generalized Gauss–Newton matrix as the principled curvature for likelihood losses (Martens, 2020). More recent optimizers such as SOAP (Vyas et al., 2025) and Muon (Liu et al., 2025) precondition LLM pretraining with matrix-level structure. AdaHessian (Yao et al., 2021) and Sophia (Liu et al., 2024) instead use a diagonal Hessian estimate, and Sophia shows that a clipped, EMA-smoothed estimate accelerates LLM pretraining, but these estimators require extra backward passes on the training node. The closest systems precedent is Distributed Shampoo (Anil et al., 2020), which computes the inverse roots of its preconditioners asynchronously on otherwise underused CPUs and refreshes them only every few hundred steps. Its statistics, however, are still gathered from backpropagated gradients on the accelerators, and only the matrix roots are offloaded. ZO-Curv moves the curvature statistic itself to forward-only devices. It does not propose a new preconditioner: it matches SophiaG step for step (Appendix C).

Training on heterogeneous and idle compute. SWARM parallelism (Ryabinin et al., 2023) and DiLoCo (Douillard et al., 2023) train across poorly connected devices, Petals (Borzunov et al., 2023) runs collaborative inference and fine-tuning of large models, and Bamboo (Thorpe et al., 2023) trains on preemptible instances. Their training workers must all run backpropagation. On the serving side, inference engines such as vLLM (Kwon et al., 2023) and DistServe (Zhong et al., 2024) are built around forward passes, and Splitwise (Patel et al., 2024) and Mélange (Griggs et al., 2024) make inference clusters more efficient. Lyra (Li et al., 2023) loans idle inference servers to training jobs during low-traffic periods, Valve (Liu et al., 2026a) colocates offline work with online serving, and Runway reclaims idle inference GPUs for research jobs (Dorsey & Kafonek, 2026). These systems hand idle serving capacity to other workloads, and loaned servers must run full training. ZO-Curv instead lets forward-only capacity accelerate the same pretraining run, without backpropagation on the serving devices.

B EXPERIMENTAL DETAILS

Table 2: **Model configurations and peak learning rates.** ZO-Curv uses the SophiaG hyperparameters unchanged; γ is the curvature scale of equation 3.

Model	d_{model}	Heads	Depth	AdamW lr	SophiaG / ZO-Curv lr	γ
GPT-2 Small (125M)	768	12	12	6×10^{-4}	6×10^{-4}	0.05
GPT-2 Medium (355M)	1024	16	24	3×10^{-4}	7×10^{-4}	0.08
GPT-2 Large (770M)	1280	20	36	2×10^{-4}	3×10^{-4}	0.05
GPT-NeoX (1.5B)	1536	24	48	1.5×10^{-4}	1.2×10^{-4}	0.05

Models and data. We build on the nanoGPT implementation used by Sophia (Liu et al., 2024): decoder-only transformers with GELU activations, no bias terms, and no dropout, with the configurations of Table 2. For the 355M and larger models we scale attention by the inverse of the layer index, as Sophia does to keep AdamW stable at these sizes. All models are trained from scratch on OpenWebText (Gokaslan & Cohen, 2019) with the GPT-2 tokenizer and the train and validation split of nanoGPT (9B and 4.4M tokens), at context length 1024 and global batch size 512, in bfloat16.

Optimization. All methods use a cosine learning-rate schedule with 2K warmup steps, decaying to a minimum between 10^{-5} and 6×10^{-5}, and gradient clipping at norm 1.0. AdamW uses $\beta = (0.9, 0.95)$ and weight decay 0.1. ZO-Curv and SophiaG use $\beta = (0.965, 0.99)$, weight decay 0.2, clipping threshold $\rho = 1$, and the curvature scale γ of Table 2, following Sophia, and refreshes its curvature at every step. Step budgets are given in Section 4.1.

Curvature estimator. Unless stated otherwise, ZO-Curv uses $q = 512$ queries per refresh with finite-difference radius $\varepsilon = 10^{-4}$, and samples the pseudo-labels from the model's own predictive distribution, as in the GNB estimator. Following the Sophia implementation, the scale B in $h^{\text{ZO}} = B\,\bar{z} \odot \bar{z}$ counts the tokens of the curvature batch, and the denominator floor is $\epsilon_0 = 10^{-15}$.

Profiling. The measurements of Section 2.3 report peak reserved GPU memory at batch size 8, with forward-only execution in fp16 and training in mixed precision, and per-step time averaged over 100 steps after warmup.

C STEP-AXIS COMPARISON WITH SOPHIAG

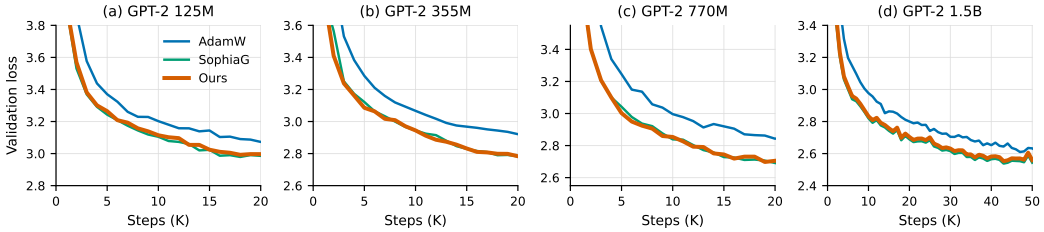

Figure 10: **ZO-Curv matches SophiaG's step efficiency at every scale** (first 20K steps of the runs of Figure 5). The two curves are nearly indistinguishable on a step axis: the forward-only estimator matches the backpropagated one in per-step quality, and the advantage of ZO-Curv lies in where the estimate is computed.

Figure 10 replots the runs of Figure 5 on a step axis. ZO-Curv and SophiaG are nearly indistinguishable at every size, confirming that the forward-only estimator matches the backpropagated GNB estimator in per-step optimization quality. The contribution of ZO-Curv is not a stronger preconditioner but a preconditioner obtained from hardware that could not otherwise participate in training, which is why the two methods separate on the wall-clock axis of Figure 5.

D SOPHIAG WITH A TEN-STEP REFRESH INTERVAL

Our main comparison refreshes the curvature of SophiaG at every step, so that it consumes curvature as fresh as that of ZO-Curv (Section 4.1). The original Sophia instead refreshes every $k = 10$ steps to amortize the cost of the GNB estimator. Figure 11 reports the time each method needs for a fixed budget of 20K steps under this setting, using the per-step measurements behind Figure 5. A ten-step interval brings SophiaG to 9–13% above AdamW per step, while ZO-Curv stays within 1–5% of AdamW because its estimate is computed off the training node. SophiaG therefore remains 6–12% slower than ZO-Curv at every size, and it obtains this reduction by refreshing its curvature ten times less often, whereas ZO-Curv refreshes at every step.

The staler curvature also costs accuracy (Figure 12). After 20K steps, SophiaG with a ten-step interval ends 0.03–0.04 above its every-step counterpart in the main runs and 0.02–0.04 above ZO-Curv at every size. Amortizing the estimator on the training node thus leaves SophiaG both slower and less accurate than ZO-Curv, which keeps the curvature fresh at nearly the per-step cost of AdamW.

E TRAINING TRAJECTORIES ACROSS QUERY BUDGETS

Figure 13 extends the trajectory panels of Figure 7(c,d) to all four model sizes. The qualitative picture is uniform: every query budget separates from AdamW early, and increasing q lowers the entire trajectory. The sizes differ only in where the budgets stop mattering, consistent with the saturation points of Figure 7(a,b).

F EFFECT OF THE ONE-STEP PIPELINE OFFSET

The pipeline of Section 3.1 applies the curvature estimated at θ_{t-1} to the update at step t. Figure 14 compares this deployed variant against a synchronous control that waits for the curvature of the current snapshot θ_t. The two loss curves coincide throughout training: consecutive snapshots differ by a single clipped update, and the curvature EMA smooths over far longer horizons than one step, so the offset costs nothing while removing the serving pool from the critical path.

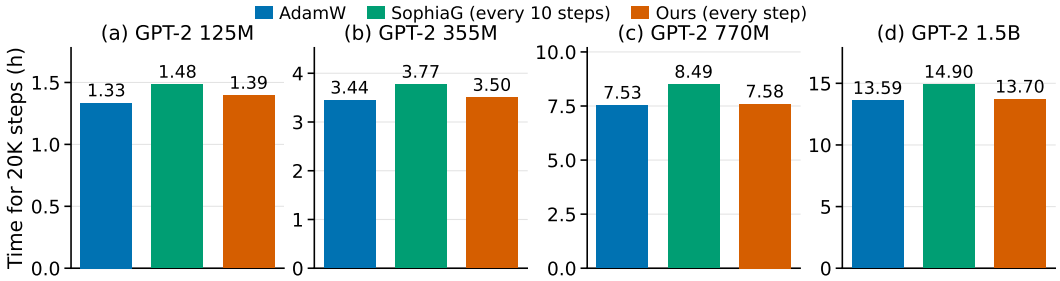

Figure 11: **SophiaG with a ten-step refresh interval.** Wall-clock time to complete 20K steps for AdamW, SophiaG refreshing its curvature every ten steps, and ZO-Curv refreshing at every step. Even with its curvature cost amortized, SophiaG is slower than ZO-Curv at every size.

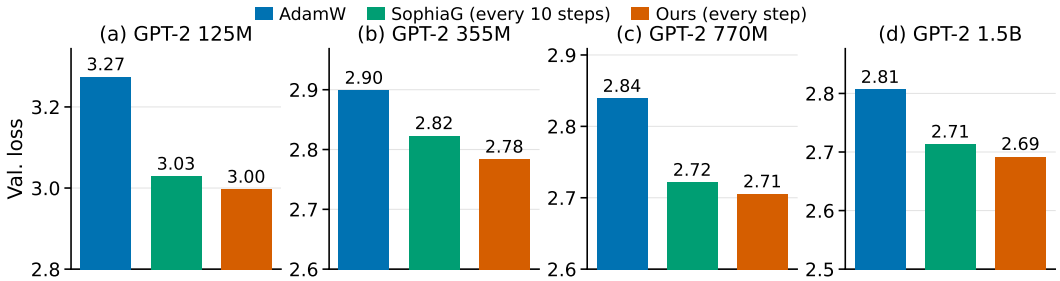

Figure 12: **Loss of SophiaG with a ten-step refresh interval.** Validation loss after 20K steps for the runs of Figure 11. With its curvature refreshed ten times less often, SophiaG ends above ZO-Curv at every size.

G Full downstream results

Table 3: **Full downstream evaluation** of the GPT-2 770M checkpoints at 20K steps (companion to Figure 9). Accuracy in %. Base: few-shot before finetuning (10 examples for HellaSwag, 25 for ARC-E and ARC-C, none for LAMBADA). FT: zero-shot after per-task LoRA finetuning. LAMBADA has no public training split and is not finetuned; it enters the FT average with its untuned score.

	HellaSwag		ARC-E		ARC-C		LAMBADA		Average	
Method	Base	FT	Base	FT	Base	FT	Base	FT	Base	FT
AdamW	31.4	33.0	40.2	50.9	22.1	26.1	30.8	–	31.1	35.2
SophiaG	35.6	37.1	47.5	54.7	25.8	28.1	39.1	–	37.0	39.8
ZO-Curv (ours)	35.2	36.3	47.4	54.2	26.8	28.8	39.0	–	37.1	39.6

Table 3 lists the per-task accuracies behind Figure 9. We evaluate the 770M checkpoint of each method after 20K steps, the shared budget of Table 1.

Before finetuning. We use few-shot prompting with 10 in-context examples for HellaSwag, 25 for ARC-Easy and ARC-Challenge, and none for LAMBADA, drawn from each training split with a fixed seed. For the three multiple-choice tasks, the prediction is the choice with the highest average log-likelihood per answer token, evaluated on the validation split (10,042, 570, and 299 examples). LAMBADA is scored by whether greedy decoding recovers the final word, on the 5,153 test examples of the OpenAI version.

After finetuning. Starting from each checkpoint, we finetune separately on the training split of every multiple-choice task with LoRA (rank 8, $\alpha = 16$, dropout 0.05) on the attention and MLP projections, which trains 5.9M parameters (0.76% of the model). We use AdamW with learning rate

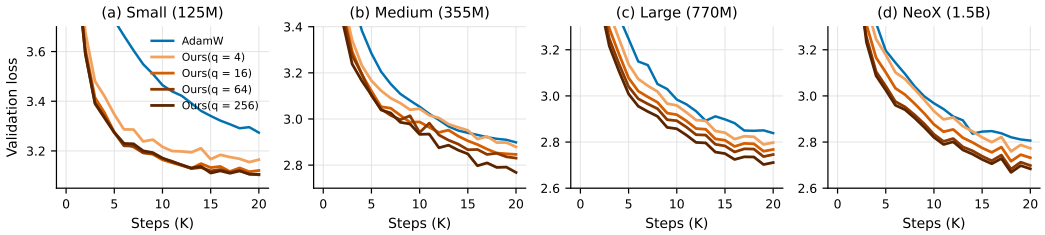

Figure 13: **Training trajectories across query budgets, all four sizes** (companion to Figure 7). At 125M the budgets collapse onto each other early, whereas at 770M and 1.5B they remain cleanly ordered, matching the scale-dependent saturation of Figure 7(a,b).

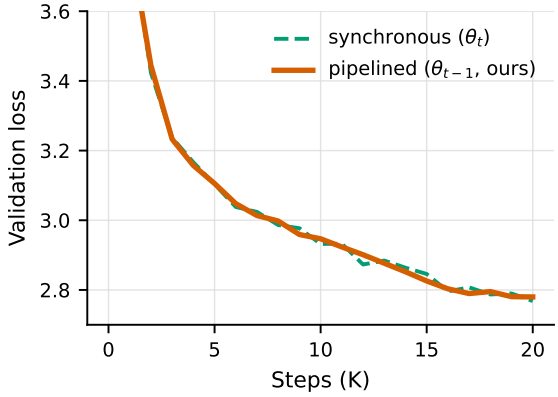

Figure 14: **The one-step pipeline offset is free** (GPT-2 355M, $q = 512$). Training with curvature estimated at θ_{t-1} (pipelined, as deployed) is indistinguishable from using curvature estimated at the current θ_t (synchronous).

2×10^{-5} and batch size 8 for 2,000 steps, with the loss restricted to the answer tokens, and then evaluate on the validation split. LAMBADA has no public training split, so it is not finetuned.

Results. Before finetuning, ZO-Curv improves over AdamW on every task, by 6.0 points on average (37.1 versus 31.1). It stays within 0.4 points of SophiaG on HellaSwag, ARC-Easy, and LAMBADA, and is 1.0 point higher on ARC-Challenge. Finetuning helps the AdamW checkpoint most, by 4.1 points on average against 2.8 for SophiaG and 2.5 for ZO-Curv, with the largest gain on ARC-Easy (10.7 points). It nonetheless trails both second-order checkpoints on every task after finetuning, while ZO-Curv remains within 0.8 points of SophiaG throughout (39.6 versus 39.8 on average).

H PROOFS

This appendix provides the full theoretical analysis of ZO-Curv. Our goal is to isolate the effect of replacing the backpropagation-based Gauss–Newton–Bartlett (GNB) curvature estimator in SophiaG with a forward-only zeroth-order estimator. The analysis proceeds in three stages. First, we identify what the average-then-square ZO estimator estimates: under a finite query budget q, it is an unbiased estimate of a damped GNB diagonal whose damping decreases as $1/q$, and its fluctuation is of the order of this target in every coordinate, independently of the dimension d. Second, we show that the curvature EMA averages these fluctuations into a bounded multiplicative perturbation of the damped reference preconditioner, and that such a perturbation does not destroy descent: the update direction remains a descent direction and converges to stationarity. Third, we combine the two results into a convergence guarantee relative to the damped-GNB-preconditioned reference update.

As in the theoretical analysis of Sophia (Liu et al., 2024), we analyze a deterministic clipped preconditioned core in order to isolate the role of curvature estimation. In particular, we omit gradient momentum and decoupled weight decay from the convergence proof. The curvature EMA is retained and updated at every iteration, matching the per-step estimator in Section 3. We first analyze the synchronous core, in which the update at step t uses the curvature queried at θ_t; Remark 2 shows that the results carry over unchanged to the pipelined update with any staleness s.

H.1 Notation and reference curvature estimator

Let

$$F(\theta) \tag{7}$$

denote the training objective, with parameters $\theta \in \mathbb{R}^d$. At iteration t, let

$$g_t := \nabla F(\theta_t) \tag{8}$$

denote the first-order gradient used to update the model.

The curvature estimator is computed from a sampled-label mini-batch, as in the GNB estimator of Sophia-G. Given a mini-batch $\{x_b\}_{b=1}^{B}$, let

$$\hat{y}_b \sim \mathrm{softmax}(f(\theta_t, x_b)), \qquad b = 1, \dots, B, \tag{9}$$

and define the sampled-label loss

$$\ell_t^{\mathrm{samp}}(\theta) := \frac{1}{B} \sum_{b=1}^{B} \ell_{\mathrm{ce}}\big(f(\theta, x_b), \hat{y}_b\big). \tag{10}$$

We denote its gradient at θ_t by

$$c_t := \nabla \ell_t^{\mathrm{samp}}(\theta_t). \tag{11}$$

Conditioned on the sampled mini-batch and pseudo-labels, c_t is fixed throughout the q ZO queries.

The backpropagation-based GNB estimator used as our reference is

$$h_t^{\mathrm{G}} := B\, c_t \odot c_t, \tag{12}$$

where $\odot$ denotes element-wise multiplication. Taking expectation over the sampled labels recovers the diagonal of the mini-batch Gauss–Newton matrix, as established by the Bartlett identities in (Liu et al., 2024). Our analysis below compares the ZO estimate with this reference conditioned on the same mini-batch and the same sampled labels.

A finite query budget adds a uniform damping to this target (Proposition 1). We therefore define the damped GNB diagonal

$$\tilde{h}_t := \left(1 + \frac{1}{q}\right) h_t^{\mathrm{G}} + \lambda_t \mathbf{1}, \qquad \lambda_t := \frac{\|h_t^{\mathrm{G}}\|_1}{q} = \frac{B\|c_t\|_2^2}{q}, \tag{13}$$

where $\mathbf{1} \in \mathbb{R}^d$ is the all-ones vector. This is a Levenberg–Marquardt-type damping of the GNB diagonal, of the kind commonly added to curvature preconditioners for numerical stability. Here it is not a tuned hyperparameter but is induced by the finite budget, and $\tilde{h}_t \to h_t^{\mathrm{G}}$ as $q \to \infty$. Adding a constant to every coordinate preserves the ordering of the coordinates by curvature.

The reference is coupled to the ZO-Curv trajectory: at every iteration, $\tilde{h}_t$ is evaluated at the ZO-Curv iterate θ_t, on the same mini-batch and with the same sampled labels as the ZO queries. It is never computed by the algorithm and serves only as a comparison point. Define the reference curvature EMA, initialized at zero as in ZO-Curv, as

$$\tilde{v}_t = \beta_2 \tilde{v}_{t-1} + (1 - \beta_2)\tilde{h}_t, \qquad \tilde{v}_0 = 0. \tag{14}$$

The corresponding reference denominator is

$$d_{t,i} := \max\{\gamma \tilde{v}_{t,i}, \epsilon_0\}, \qquad D_t := \mathrm{Diag}(d_t), \tag{15}$$

where $\epsilon_0 > 0$ is the numerical denominator floor. We use ε exclusively for the ZO finite-difference radius, to distinguish it from ϵ_0.

H.2 ZO APPROXIMATION OF THE GNB CURVATURE

At each iteration, ZO-Curv draws q independent Gaussian perturbations

$$u_{t,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d), \qquad j = 1, \dots, q. \tag{16}$$

Importantly, all q queries use the same mini-batch and the same sampled pseudo-labels, and hence share the same curvature gradient c_t.

For direction $u_{t,j}$, define the central-difference directional derivative

$$\Delta_{t,j}^{\varepsilon} := \frac{\ell_t^{\text{samp}}(\theta_t + \varepsilon u_{t,j}) - \ell_t^{\text{samp}}(\theta_t - \varepsilon u_{t,j})}{2\varepsilon}. \tag{17}$$

The corresponding ZO gradient estimate is

$$z_{t,j}^{\varepsilon} := \Delta_{t,j}^{\varepsilon} u_{t,j}. \tag{18}$$

ZO-Curv first averages the q gradient estimates,

$$\bar{z}_t^{\varepsilon} := \frac{1}{q} \sum_{j=1}^{q} z_{t,j}^{\varepsilon}, \tag{19}$$

and then squares the averaged estimate:

$$h_t^{\text{ZO}} := B\, \bar{z}_t^{\varepsilon} \odot \bar{z}_t^{\varepsilon}. \tag{20}$$

We first study the ideal directional-derivative limit $\varepsilon \to 0$.

Lemma 1 (Moments of the averaged ZO gradient)**.** Fix a vector $c \in \mathbb{R}^d$ and let $u_1, \dots, u_q \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d)$. Define

$$z_j := (c^\top u_j) u_j, \qquad \bar{z} := \frac{1}{q} \sum_{j=1}^{q} z_j. \tag{21}$$

Then, for every coordinate i,

$$\mathbb{E}[\bar{z}_i \mid c] = c_i, \tag{22}$$

$$\mathbb{E}[\bar{z}_i^2 \mid c] = \left(1 + \frac{1}{q}\right) c_i^2 + \frac{1}{q} \|c\|_2^2, \tag{23}$$

$$\mathrm{Var}(\bar{z}_i \mid c) = \frac{\|c\|_2^2 + c_i^2}{q}. \tag{24}$$

Proof. For a single Gaussian direction,

$$\mathbb{E}[(c^\top u) u_i] = \sum_{r=1}^{d} c_r \mathbb{E}[u_r u_i] = c_i, \tag{25}$$

which proves equation 22.

For the second moment, Isserlis' identity for a standard Gaussian gives

$$\mathbb{E}\left[(c^\top u)^2 u_i^2\right] = \|c\|_2^2 + 2c_i^2. \tag{26}$$

Therefore

$$\mathrm{Var}\left((c^\top u) u_i\right) = \|c\|_2^2 + c_i^2. \tag{27}$$

Averaging q independent directions divides the variance by q, hence

$$\mathrm{Var}(\bar{z}_i \mid c) = \frac{\|c\|_2^2 + c_i^2}{q}. \tag{28}$$

Combining this with $\mathbb{E}[\bar{z}_i \mid c] = c_i$ yields

$$\mathbb{E}[\bar{z}_i^2 \mid c] = c_i^2 + \frac{\|c\|_2^2 + c_i^2}{q} \tag{29}$$

$$= \left(1 + \frac{1}{q}\right) c_i^2 + \frac{1}{q} \|c\|_2^2. \tag{30}$$

□

Lemma 1 shows that the averaged ZO gradient itself approaches c at the canonical $q^{-1/2}$ stochastic rate. Its second moment equation 23 is the source of the damping in equation 13. For the convergence analysis, however, we require a high-probability rather than an expectation bound.

Lemma 2 (High-probability ZO gradient error)**.** Under the setting of Lemma 1, there exists a universal constant $C_0 > 0$ such that, for any failure probability $\zeta \in (0, 1)$, with probability at least $1 - \zeta$,

$$\|\bar{z} - c\|_\infty \le C_0 \|c\|_2 \left[\sqrt{\frac{\log(2d/\zeta)}{q}} + \frac{\log(2d/\zeta)}{q} \right]. \tag{31}$$

Proof. Fix a coordinate i and define

$$X_{j,i} := (c^\top u_j) u_{j,i} - c_i. \tag{32}$$

We have $\mathbb{E}[X_{j,i}] = 0$. Moreover, $c^\top u_j$ is a centered Gaussian with variance $\|c\|_2^2$, and $u_{j,i}$ is standard Gaussian. The product of two sub-Gaussian random variables is sub-exponential, even when the two variables are correlated. Therefore, for a universal constant $C > 0$,

$$\|X_{j,i}\|_{\psi_1} \le C \|c\|_2. \tag{33}$$

Bernstein's inequality for independent centered sub-exponential variables then gives

$$\Pr\left[\left| \frac{1}{q} \sum_{j=1}^{q} X_{j,i} \right| > C_0 \|c\|_2 \left(\sqrt{\frac{s}{q}} + \frac{s}{q} \right) \right] \le 2e^{-s}. \tag{34}$$

Setting

$$s = \log(2d/\zeta) \tag{35}$$

and applying a union bound over $i = 1, \dots, d$ proves equation 31. □

We next account for the finite central-difference radius ε.

Assumption A1 (Local Hessian Lipschitzness)**.** For every iteration t, the sampled-label loss ℓ_t^{samp} is twice differentiable with an L_H-Lipschitz Hessian on the ball $\mathcal{U}_t := \{\theta : \|\theta - \theta_t\|_2 \le \varepsilon R\}$ containing all ZO query points:

$$\|\nabla^2 \ell_t^{\mathrm{samp}}(\theta) - \nabla^2 \ell_t^{\mathrm{samp}}(\theta')\|_2 \le L_H \|\theta - \theta'\|_2, \qquad \theta, \theta' \in \mathcal{U}_t. \tag{36}$$

The query points are $\theta_t \pm \varepsilon u_{t,j}$ with $\|u_{t,j}\|_2 \approx \sqrt{d}$, so the relevant radius is $\varepsilon \max_j \|u_{t,j}\|_2$ rather than ε; we take $R = R_q$ in Proposition 1 and $R = R_T$ in Theorem 3, where R_q and R_T bound $\max_j \|u_{t,j}\|_2$ with high probability.

Lemma 3 (Central-difference error)**.** Under Assumption A1, for any direction u with $\|u\|_2 \le R$,

$$\left| \frac{\ell_t^{\mathrm{samp}}(\theta_t + \varepsilon u) - \ell_t^{\mathrm{samp}}(\theta_t - \varepsilon u)}{2\varepsilon} - c_t^\top u \right| \le \frac{L_H \varepsilon^2}{6} \|u\|_2^3. \tag{37}$$

Proof. Define the scalar function

$$\phi(s) := \ell_t^{\mathrm{samp}}(\theta_t + su), \qquad |s| \le \varepsilon. \tag{38}$$

Then $\phi'(0) = c_t^\top u$ and $\phi''(s) = u^\top \nabla^2 \ell_t^{\mathrm{samp}}(\theta_t + su) u$, so Assumption A1 gives

$$|\phi''(s) - \phi''(0)| \le L_H |s| \|u\|_2^3. \tag{39}$$

Taylor's theorem with integral remainder, $\phi(\pm\varepsilon) = \phi(0) \pm \varepsilon\phi'(0) + \int_0^{\pm\varepsilon} (\pm\varepsilon - s)\phi''(s)\, ds$, then yields

$$\left| \phi(\pm\varepsilon) - \phi(0) \mp \varepsilon\phi'(0) - \frac{\varepsilon^2}{2}\phi''(0) \right| \le L_H \|u\|_2^3 \int_0^\varepsilon (\varepsilon - s) s \, ds = \frac{L_H \varepsilon^3}{6} \|u\|_2^3. \tag{40}$$

Subtracting the two expansions cancels $\phi(0)$ and $\phi''(0)$, and dividing by 2ε gives

$$\left| \frac{\phi(\varepsilon) - \phi(-\varepsilon)}{2\varepsilon} - \phi'(0) \right| \le \frac{L_H \varepsilon^2}{6} \|u\|_2^3. \tag{41}$$

This proves the result; no third derivative is required. □

We now combine Lemmas 1–3 into the properties of the curvature estimate used in the rest of the analysis. Write

$$h_t^{\mathrm{ZO},0} := B\,\bar{z}_t^0 \odot \bar{z}_t^0, \qquad \bar{z}_t^0 := \frac{1}{q}\sum_{j=1}^{q}(c_t^\top u_{t,j})u_{t,j}, \tag{42}$$

for the ideal estimate with exact directional derivatives, and h_t^{ZO} for the central-difference estimate equation 20.

Proposition 1 (Forward-only estimation of damped GNB curvature)**.** Fix an iteration t and condition on its mini-batch and sampled labels, with $c_t \neq 0$.

(i) *Unbiasedness.* The ideal estimate is unbiased for the damped target equation 13:

$$\mathbb{E}\big[h_t^{\mathrm{ZO},0}\big] = \tilde{h}_t. \tag{43}$$

(ii) *Relative second moment.* There is a universal constant $C_v \geq 1$ such that, for every coordinate i,

$$\mathbb{E}\big[(h_{t,i}^{\mathrm{ZO},0})^2\big] \leq C_v \tilde{h}_{t,i}^2. \tag{44}$$

(iii) *Relative envelope and finite differences.* For $\zeta \in (0,1)$, define

$$\begin{aligned} \Lambda &:= \log\frac{4d}{\zeta}, \qquad R_q := \sqrt{d} + \sqrt{2\log\frac{4q}{\zeta}}, \\ M_{q,d} &:= \sqrt{2\log\frac{8qd}{\zeta}}, \qquad \vartheta := \frac{L_H\varepsilon^2}{6}R_q^3 M_{q,d}, \end{aligned} \tag{45}$$

and suppose $q \geq \Lambda$ and that Assumption A1 holds with $R = R_q$. Then, with probability at least $1-\zeta$, simultaneously for all coordinates i,

$$h_{t,i}^{\mathrm{ZO},0} \leq C_b \Lambda\, \tilde{h}_{t,i}, \qquad C_b := \max\{2, 8C_0^2\}, \tag{46}$$

$$\big|h_{t,i}^{\mathrm{ZO}} - h_{t,i}^{\mathrm{ZO},0}\big| \leq 2\sqrt{C_b \Lambda B \vartheta^2\, \tilde{h}_{t,i}} + B\vartheta^2, \tag{47}$$

where C_0 is the constant of Lemma 2.

Proof. For (i), Lemma 1 with $c = c_t$ gives

$$\mathbb{E}\big[h_{t,i}^{\mathrm{ZO},0}\big] = B\,\mathbb{E}\big[(\bar{z}_{t,i}^0)^2\big] = \left(1 + \frac{1}{q}\right) B c_{t,i}^2 + \frac{B\|c_t\|_2^2}{q} = \tilde{h}_{t,i}. \tag{48}$$

For (ii), write $\bar{z}_{t,i}^0 = c_{t,i} + a_i$ with $a_i := q^{-1}\sum_{j=1}^q X_{j,i}$ and $X_{j,i}$ as in the proof of Lemma 2. Since $(x+y)^4 \leq 8(x^4+y^4)$,

$$\mathbb{E}\big[(\bar{z}_{t,i}^0)^4\big] \leq 8c_{t,i}^4 + 8\,\mathbb{E}[a_i^4]. \tag{49}$$

The $X_{j,i}$ are independent and centered, so

$$\mathbb{E}[a_i^4] = \frac{q\,\mathbb{E}[X_{1,i}^4] + 3q(q-1)\big(\mathbb{E}[X_{1,i}^2]\big)^2}{q^4} \leq \frac{\mathbb{E}[X_{1,i}^4] + 3\big(\mathbb{E}[X_{1,i}^2]\big)^2}{q^2}. \tag{50}$$

By Lemma 1, $\mathbb{E}[X_{1,i}^2] = \|c_t\|_2^2 + c_{t,i}^2 \leq 2\|c_t\|_2^2$, and since $\|X_{1,i}\|_{\psi_1} \leq C\|c_t\|_2$, also $\mathbb{E}[X_{1,i}^4] \leq C'\|c_t\|_2^4$ for a universal constant C'. Hence $\mathbb{E}[a_i^4] \leq C''\|c_t\|_2^4/q^2$ with $C'' := C' + 12$, and

$$\mathbb{E}\big[(h_{t,i}^{\mathrm{ZO},0})^2\big] = B^2\,\mathbb{E}\big[(\bar{z}_{t,i}^0)^4\big] \leq 8\big(Bc_{t,i}^2\big)^2 + 8C''\lambda_t^2 \tag{51}$$

$$\leq C_v\big(Bc_{t,i}^2 + \lambda_t\big)^2 \leq C_v \tilde{h}_{t,i}^2, \tag{52}$$

with $C_v := 8\max\{1, C''\}$, where the last step uses $Bc_{t,i}^2 + \lambda_t \leq \tilde{h}_{t,i}$.

For (iii), we split the failure probability ζ into three events. First, Lemma 2 with failure probability $\zeta/2$, for which $\log(2d/(\zeta/2)) = \Lambda$, gives $|a_i| \le C_0\|c_t\|_2(\sqrt{\Lambda/q} + \Lambda/q)$ for all i. Since $q \ge \Lambda$ and $(\sqrt{x} + x)^2 \le 4x$ for $x \le 1$,

$$Ba_i^2 \le 4C_0^2\Lambda\frac{B\|c_t\|_2^2}{q} = 4C_0^2\Lambda\lambda_t. \tag{53}$$

Therefore, using $\Lambda \ge 1$,

$$h_{t,i}^{\mathrm{ZO},0} = B(c_{t,i} + a_i)^2 \le 2Bc_{t,i}^2 + 8C_0^2\Lambda\lambda_t \le C_b\Lambda\,\tilde{h}_{t,i}, \tag{54}$$

which is equation 46.

Second, Gaussian norm concentration, $\Pr[\|u\|_2 \ge \sqrt{d} + s] \le e^{-s^2/2}$, and a union bound over the q directions give, with probability at least $1 - \zeta/4$, $\max_{j\le q}\|u_{t,j}\|_2 \le R_q$. Third, the Gaussian tail bound $\Pr[|u_{j,i}| \ge s] \le 2e^{-s^2/2}$ and a union bound over all qd coordinates give, with probability at least $1 - \zeta/4$, $\max_{j\le q,\, i\le d}|u_{t,j,i}| \le M_{q,d}$. On the second event every query point lies in $\mathcal{U}_t$ with $R = R_q$, so Lemma 3 applies to every direction and

$$\bar{z}_t^{\varepsilon} - \bar{z}_t^{0} = \frac{1}{q}\sum_{j=1}^{q} e_{t,j}^{\mathrm{FD}}, \qquad |e_{t,j,i}^{\mathrm{FD}}| \le \frac{L_H\varepsilon^2}{6}\|u_{t,j}\|_2^3|u_{t,j,i}|. \tag{55}$$

On both events, $e_i := \bar{z}_{t,i}^{\varepsilon} - \bar{z}_{t,i}^{0}$ satisfies $|e_i| \le \vartheta$. Then

$$\left|h_{t,i}^{\mathrm{ZO}} - h_{t,i}^{\mathrm{ZO},0}\right| = B\left|2\bar{z}_{t,i}^{0}e_i + e_i^2\right| \le 2\sqrt{B\,h_{t,i}^{\mathrm{ZO},0}}\,\vartheta + B\vartheta^2, \tag{56}$$

and equation 46 gives equation 47. A union bound over the three events, with total failure probability $\zeta/2 + \zeta/4 + \zeta/4 = \zeta$, completes the proof. □

Proposition 1 shows that, like the GNB estimator of Sophia-G, the forward-only estimate is unbiased; its target is the GNB diagonal with the additional uniform damping $\lambda_t = \|h_t^{\mathrm{G}}\|_1/q$. Averaging before squaring is what keeps this damping small: squaring each directional estimate before averaging would give $\mathbb{E}[B(\Delta_{t,j}^0 u_{t,j,i})^2] = 2h_{t,i}^{\mathrm{G}} + \|h_t^{\mathrm{G}}\|_1$, a damping q times larger that does not decrease with the budget. Moreover, by (ii) and (iii), the fluctuation of the estimate around its target is at most of the order of the target itself in every coordinate, with constants independent of d, q, and c_t. This relative control is what the curvature EMA exploits next. Dividing equation 47 by $\tilde{h}_{t,i}$ shows that the finite-difference perturbation is also relative, of order $\sqrt{\Lambda B\vartheta^2/\tilde{h}_{t,i}} = O(\varepsilon^2)$.

H.3 Effect of curvature EMA

The optimizer does not use a single raw curvature estimate directly. Instead, both Sophia-G and ZO-Curv maintain a curvature EMA. Because fresh directions are drawn at every refresh, the EMA averages the relative fluctuations of Proposition 1 over its effective window $1/(1 - \beta_2)$.

At every iteration, define

$$v_t^{\mathrm{ZO}} = \beta_2 v_{t-1}^{\mathrm{ZO}} + (1 - \beta_2)h_t^{\mathrm{ZO}}, \qquad v_0^{\mathrm{ZO}} = 0, \tag{57}$$

and let $w_{t,s} := (1 - \beta_2)\beta_2^{t-s}$ and $W_t := \sum_{s=1}^{t} w_{t,s} = 1 - \beta_2^t$, so that $v_t^{\mathrm{ZO}} = \sum_{s\le t} w_{t,s}h_s^{\mathrm{ZO}}$ and $\tilde{v}_t = \sum_{s\le t} w_{t,s}\tilde{h}_s$. Let $\mathcal{G}_s$ denote the history up to and including θ_s and the step-s mini-batch and sampled labels, but not the step-s directions. Then c_s and $\tilde{h}_s$ are $\mathcal{G}_s$-measurable, the directions $u_{s,1}, \ldots, u_{s,q}$ are independent of $\mathcal{G}_s$, and h_s^{ZO} is $\mathcal{G}_{s+1}$-measurable.

Assumption A2 (Slowly varying damped curvature)**.** Over the analyzed horizon $s \le T$, there exist $\kappa \ge 1$ and deterministic $\bar{H}_i > 0$ such that

$$\frac{\bar{H}_i}{\kappa} \le \tilde{h}_{s,i} \le \bar{H}_i, \qquad s \le T,\ i = 1, \ldots, d. \tag{58}$$

almost surely along the ZO-Curv trajectory.

Since $\tilde{h}_{s,i} \geq \lambda_s$, the damping keeps every coordinate away from zero, and the assumption only asks that the damped curvature of each coordinate varies by at most a factor κ over the horizon. Let $\underline{H} := \min_i \bar{H}_i$.

Lemma 4 (EMA concentration)**.** Suppose Assumptions A1 and A2 hold over T iterations. For $\zeta \in (0,1)$, define

$$\begin{aligned} \Lambda_T &:= \log \frac{16qdT}{\zeta}, \qquad R_T := \sqrt{d} + \sqrt{2\log\frac{8qT}{\zeta}}, \\ \vartheta_T &:= \frac{L_H \varepsilon^2}{6} R_T^3 \sqrt{2\Lambda_T}, \qquad \varrho_T := \frac{\kappa B \vartheta_T^2}{\underline{H}}, \end{aligned} \tag{59}$$

take $R = R_T$ in Assumption A1, and suppose $q \geq \Lambda_T$. Let $t_0 := \lceil \ln 2/(1-\beta_2) \rceil$ and

$$\delta_T := 2\kappa \left[\sqrt{2C_v(1-\beta_2)\Lambda_T} + \frac{4}{3} C_b (1-\beta_2)\Lambda_T^2 \right] + \sqrt{\frac{C_v \zeta}{2T}} + 2\sqrt{C_b \Lambda_T \varrho_T} + \varrho_T. \tag{60}$$

Then, with probability at least $1-\zeta$, simultaneously for all $t_0 \leq t \leq T$ and all coordinates i,

$$\left| v_{t,i}^{\mathrm{ZO}} - \tilde{v}_{t,i} \right| \leq \delta_T \, \tilde{v}_{t,i}. \tag{61}$$

Proof. For each $s \leq T$, let $\mathcal{E}_s$ be the event of Proposition 1(iii) at step s with failure probability $\zeta/(2T)$. Its constants satisfy $\log(8dT/\zeta) \leq \Lambda_T$, $R_q = R_T$, and $M_{q,d} = \sqrt{2\log(16qdT/\zeta)} = \sqrt{2\Lambda_T}$, so the bounds of Proposition 1(iii) hold with Λ_T and ϑ_T. Because the step-s directions are independent of $\mathcal{G}_s$, the proposition applies conditionally on $\mathcal{G}_s$, and $\Pr[\mathcal{E}_s^c \mid \mathcal{G}_s] \leq \zeta/(2T)$.

Fix a coordinate i and decompose

$$v_{t,i}^{\mathrm{ZO}} - \tilde{v}_{t,i} = \sum_{s \leq t} w_{t,s} \big(h_{s,i}^{\mathrm{ZO}} - h_{s,i}^{\mathrm{ZO},0} \big) + \sum_{s \leq t} w_{t,s} \, \xi_{s,i}, \qquad \xi_{s,i} := h_{s,i}^{\mathrm{ZO},0} - \tilde{h}_{s,i}. \tag{62}$$

Finite-difference term. On $\mathcal{E}_s$, equation 47 and $\tilde{h}_{s,i} \geq \bar{H}_i/\kappa \geq \underline{H}/\kappa$ give

$$\left| h_{s,i}^{\mathrm{ZO}} - h_{s,i}^{\mathrm{ZO},0} \right| \leq \tilde{h}_{s,i} \left[2\sqrt{\frac{C_b \Lambda_T B \vartheta_T^2}{\tilde{h}_{s,i}}} + \frac{B\vartheta_T^2}{\tilde{h}_{s,i}} \right] \leq \big(2\sqrt{C_b \Lambda_T \varrho_T} + \varrho_T \big) \tilde{h}_{s,i}, \tag{63}$$

so the first sum is at most $(2\sqrt{C_b \Lambda_T \varrho_T} + \varrho_T)\tilde{v}_{t,i}$.

Truncation. By Proposition 1(i), $\mathbb{E}[\xi_{s,i} \mid \mathcal{G}_s] = 0$. Let $b_i := C_b \Lambda_T \bar{H}_i$. Since $h_{s,i}^{\mathrm{ZO},0}$ and $\tilde{h}_{s,i}$ are nonnegative and $C_b \Lambda_T \geq 1$, on $\mathcal{E}_s$ we have $|\xi_{s,i}| \leq C_b \Lambda_T \tilde{h}_{s,i} \leq b_i$. Define the truncated martingale differences

$$\xi'_{s,i} := \xi_{s,i} \mathbb{1}\{|\xi_{s,i}| \leq b_i\} - \mathbb{E}\big[\xi_{s,i} \mathbb{1}\{|\xi_{s,i}| \leq b_i\} \mid \mathcal{G}_s\big]. \tag{64}$$

Because $\mathbb{E}[\xi_{s,i} \mid \mathcal{G}_s] = 0$, the Cauchy–Schwarz inequality and equation 44 bound the conditional mean that is removed:

$$\left| \mathbb{E}\big[\xi_{s,i} \mathbb{1}\{|\xi_{s,i}| \leq b_i\} \mid \mathcal{G}_s\big] \right| = \left| \mathbb{E}\big[\xi_{s,i} \mathbb{1}\{|\xi_{s,i}| > b_i\} \mid \mathcal{G}_s\big] \right| \leq \sqrt{C_v} \, \tilde{h}_{s,i} \sqrt{\frac{\zeta}{2T}}. \tag{65}$$

Hence, on $\bigcap_{s \leq T} \mathcal{E}_s$,

$$\left| \sum_{s \leq t} w_{t,s} \xi_{s,i} \right| \leq \left| \sum_{s \leq t} w_{t,s} \xi'_{s,i} \right| + \sqrt{\frac{C_v \zeta}{2T}} \, \tilde{v}_{t,i}. \tag{66}$$

Martingale term. The sequence $Y_s := w_{t,s} \xi'_{s,i}$ is a martingale difference sequence with respect to $(\mathcal{G}_{s+1})_s$, with $|Y_s| \leq 2(1-\beta_2) b_i$ and, by Assumption A2 and equation 44,

$$\sum_{s \leq t} \mathbb{E}\big[Y_s^2 \mid \mathcal{G}_s\big] \leq C_v \bar{H}_i^2 \sum_{s \leq t} w_{t,s}^2 \leq C_v (1-\beta_2) \bar{H}_i^2. \tag{67}$$

Freedman's inequality then gives, with probability at least $1 - \zeta/(2dT)$,

$$\left| \sum_{s \leq t} Y_s \right| \leq \bar{H}_i \left[\sqrt{2C_v(1-\beta_2)\Lambda_T} + \frac{4}{3} C_b (1-\beta_2) \Lambda_T^2 \right], \tag{68}$$

where we used $\log(4dT/\zeta) \le \Lambda_T$. Finally, for $t \ge t_0$ we have $\beta_2^t \le e^{-(1-\beta_2)t} \le 1/2$, so $W_t \ge 1/2$ and $\tilde{v}_{t,i} \ge W_t \bar{H}_i/\kappa \ge \bar{H}_i/(2\kappa)$ by Assumption A2. Dividing by $\tilde{v}_{t,i}$ and collecting the three terms gives equation 61 for this pair (t, i). A union bound over the events $\mathcal{E}_s$, with total failure probability $\zeta/2$, and over the dT Freedman events, with total failure probability $\zeta/2$, completes the proof. □

Lemma 4 bounds the error relative to the reference EMA itself. The leading term is governed by the EMA window $1/(1-\beta_2)$ and depends on d, q, and T only through Λ_T. It does not require $q \to \infty$: the query budget enters the analysis through the damping λ_t of the target rather than through the estimation error.

H.4 From curvature error to preconditioner distortion

Define the actual ZO-Curv denominator

$$\hat{d}_{t,i} := \max\{\gamma v_{t,i}^{\mathrm{ZO}}, \epsilon_0\}, \qquad \widehat{D}_t := \mathrm{Diag}(\hat{d}_t). \tag{69}$$

Lemma 5 (Denominator distortion)**.** Suppose that, for some $\delta \in [0, 1)$ and every coordinate i,

$$(1-\delta)\tilde{v}_{t,i} \le v_{t,i}^{\mathrm{ZO}} \le (1+\delta)\tilde{v}_{t,i}. \tag{70}$$

Then

$$(1-\delta)D_t \preceq \widehat{D}_t \preceq (1+\delta)D_t. \tag{71}$$

Proof. The map $x \mapsto \max\{\gamma x, \epsilon_0\}$ is nondecreasing. For the upper bound,

$$\hat{d}_{t,i} \le \max\{(1+\delta)\gamma\tilde{v}_{t,i}, \epsilon_0\} \le (1+\delta)\max\{\gamma\tilde{v}_{t,i}, \epsilon_0\} = (1+\delta)d_{t,i}. \tag{72}$$

For the lower bound, since $\epsilon_0 \ge (1-\delta)\epsilon_0$,

$$\hat{d}_{t,i} \ge \max\{(1-\delta)\gamma\tilde{v}_{t,i}, \epsilon_0\} \ge (1-\delta)\max\{\gamma\tilde{v}_{t,i}, \epsilon_0\} = (1-\delta)d_{t,i}. \tag{73}$$

Both matrices are diagonal, which proves equation 71. □

Lemma 5 is the interface between the estimator analysis and the optimization analysis. Because the error bound of Lemma 4 is relative, it passes through the floor without any division by ϵ_0, and the entire effect of the forward-only ZO estimator is summarized by the multiplicative distortion δ.

H.5 Robust descent under approximate curvature

We now analyze the deterministic clipped preconditioned update

$$\theta_{t+1} = \theta_t - \eta s_t, \tag{74}$$

where

$$s_t := \mathrm{clip}\left(\widehat{D}_t^{-1} g_t, \rho\right). \tag{75}$$

The clipping operator is applied coordinate-wise:

$$\mathrm{clip}(x, \rho) := \mathrm{sign}(x)\min\{|x|, \rho\}. \tag{76}$$

The implementation corresponds to $\rho = 1$.

The following assumption is a preconditioned analogue of the usual smoothness assumption.

Assumption A3 (Local preconditioned smoothness)**.** There exists $\alpha > 0$ such that, for every iteration t and every point on the update segment,

$$\nabla^2 F(\theta_t - \tau\eta s_t) \preceq \alpha D_t, \qquad \forall \tau \in [0, 1]. \tag{77}$$

For ordinary gradient descent, smoothness assumes $\nabla^2 F \preceq LI$. Assumption A3 replaces the isotropic matrix LI by the reference curvature preconditioner D_t. Because the reference target is damped, ordinary smoothness already implies this assumption. For $t \ge t_0$ (defined in Lemma 4), $d_{t,i} \ge \gamma\tilde{v}_{t,i} \ge \gamma W_t \min_{s\le t}\lambda_s \ge \frac{\gamma}{2}\min_{s\le t}\lambda_s$, so $\nabla^2 F \preceq LI$ along the update segments implies Assumption A3 with $\alpha = 2L/(\gamma \min_{s\le t}\lambda_s)$. This is only a sufficient condition; the assumption holds with a smaller α whenever the GNB part of D_t captures the curvature of F.

Theorem 1 (Robust descent under curvature estimation error)**.** Suppose

$$(1-\delta)D_t \preceq \widehat{D}_t \preceq (1+\delta)D_t \tag{78}$$

for some $\delta \in [0,1)$, and Assumption A3 holds. Define

$$\Psi_t := g_t^\top s_t. \tag{79}$$

Then

$$\Psi_t = \sum_{i=1}^{d} \min\left\{ \frac{g_{t,i}^2}{\hat{d}_{t,i}}, \rho|g_{t,i}| \right\} \geq 0, \tag{80}$$

and the one-step loss satisfies

$$F(\theta_{t+1}) \leq F(\theta_t) - \eta\left(1 - \frac{\alpha\eta}{2(1-\delta)}\right)\Psi_t. \tag{81}$$

Therefore, every non-stationary step is a descent step whenever

$$0 < \eta < \frac{2(1-\delta)}{\alpha}. \tag{82}$$

In particular, if

$$\eta \leq \frac{1-\delta}{\alpha}, \tag{83}$$

then

$$F(\theta_{t+1}) \leq F(\theta_t) - \frac{\eta}{2}\Psi_t. \tag{84}$$

Proof. Since every $\hat{d}_{t,i} > 0$, clipping preserves the sign of $g_{t,i}$. Hence

$$g_{t,i}s_{t,i} = |g_{t,i}| \min\left\{ \frac{|g_{t,i}|}{\hat{d}_{t,i}}, \rho \right\} \tag{85}$$

$$= \min\left\{ \frac{g_{t,i}^2}{\hat{d}_{t,i}}, \rho|g_{t,i}| \right\} \geq 0. \tag{86}$$

Summing over coordinates gives equation 80.

Taylor's theorem with integral remainder and Assumption A3 imply

$$F(\theta_t - \eta s_t) \leq F(\theta_t) - \eta g_t^\top s_t + \frac{\alpha\eta^2}{2} s_t^\top D_t s_t. \tag{87}$$

It remains to upper-bound the quadratic term. By definition of clipping,

$$|s_{t,i}| \leq \frac{|g_{t,i}|}{\hat{d}_{t,i}}. \tag{88}$$

Therefore

$$d_{t,i}s_{t,i}^2 \leq d_{t,i}|s_{t,i}|\frac{|g_{t,i}|}{\hat{d}_{t,i}} \tag{89}$$

$$= \frac{d_{t,i}}{\hat{d}_{t,i}}|g_{t,i}||s_{t,i}|. \tag{90}$$

From $\hat{d}_{t,i} \geq (1-\delta)d_{t,i}$,

$$\frac{d_{t,i}}{\hat{d}_{t,i}} \leq \frac{1}{1-\delta}. \tag{91}$$

Since $g_{t,i}$ and $s_{t,i}$ have the same sign,

$$|g_{t,i}||s_{t,i}| = g_{t,i}s_{t,i}. \tag{92}$$

Thus

$$d_{t,i}s_{t,i}^2 \le \frac{1}{1-\delta} g_{t,i}s_{t,i}. \tag{93}$$

Summing over i yields

$$s_t^\top D_t s_t \le \frac{1}{1-\delta} g_t^\top s_t = \frac{1}{1-\delta}\Psi_t. \tag{94}$$

Substituting equation 94 into equation 87 gives equation 81. The remaining claims follow immediately from the corresponding stepsize choices. □

The key property in Theorem 1 is that curvature error affects the magnitude of the update but not its coordinate-wise sign. In contrast, replacing g_t itself with a noisy ZO gradient may change the sign of the first-order descent term $g_t^\top \hat{g}_t$. This difference formalizes why noisy ZO information is more naturally tolerated as a preconditioner than as a descent direction.

Corollary 1 (Convergence to stationarity)**.** Assume the conditions of Theorem 1, let

$$\eta \le \frac{1-\delta}{\alpha}, \tag{95}$$

and suppose F is bounded below by $F_\star$. Then, for any $T \ge 1$,

$$\min_{0\le t<T} \Psi_t \le \frac{2\,(F(\theta_0) - F_\star)}{\eta T}. \tag{96}$$

Consequently,

$$\min_{0\le t<T} \Psi_t = O(T^{-1}). \tag{97}$$

Proof. Equation equation 84 gives

$$F(\theta_t) - F(\theta_{t+1}) \ge \frac{\eta}{2}\Psi_t. \tag{98}$$

Summing from $t = 0$ to $T-1$,

$$F(\theta_0) - F(\theta_T) \ge \frac{\eta}{2}\sum_{t=0}^{T-1}\Psi_t. \tag{99}$$

Since $F(\theta_T) \ge F_\star$,

$$\sum_{t=0}^{T-1}\Psi_t \le \frac{2(F(\theta_0) - F_\star)}{\eta}. \tag{100}$$

The minimum is no larger than the average, giving equation 96. □

Remark 1 (From Ψ_t to the gradient)**.** The measure Ψ_t controls the gradient whenever the denominator is bounded above. Under Assumption A2, $\tilde{v}_{t,i} \le W_t \bar{H}_i \le \bar{H}_i$, so on the event of Theorem 3, $\hat{d}_{t,i} \le (1+\delta_T) d_{t,i} \le \hat{d}_{\max} := (1+\delta_T)\max\{\gamma \max_i \bar{H}_i, \epsilon_0\}$. Since $a \mapsto \min\{a^2/\hat{d}_{\max}, \rho|a|\}$ is increasing in $|a|$, $\Psi_t \le \psi$ implies $|g_{t,i}| \le \max\{\sqrt{\hat{d}_{\max}\psi}, \psi/\rho\}$ for every coordinate i.

H.6 Local linear convergence

The global result above does not require clipping to be inactive. Once the optimization enters a sufficiently local region, however, the preconditioned gradient becomes small and clipping no longer activates. In this regime we obtain a sharper linear rate.

Assumption A4 (Preconditioned Polyak–Lojasiewicz condition)**.** There exists $\mu_D \in (0, \alpha]$ such that, whenever θ_t lies in a neighborhood $\mathcal{N}$ of the target solution,

$$g_t^\top D_t^{-1} g_t \ge 2\mu_D\,(F(\theta_t) - F_\star)\,. \tag{101}$$

Theorem 2 (Local linear convergence with approximate curvature)**.** Suppose Assumptions A3 and A4 hold, and

$$(1-\delta)D_t \preceq \widehat{D}_t \preceq (1+\delta)D_t, \qquad 0 \le \delta < 1. \tag{102}$$

Suppose further that

$$\|D_t^{-1} g_t\|_\infty \le (1-\delta)\rho. \tag{103}$$

Then clipping is inactive and

$$s_t = \widehat{D}_t^{-1} g_t. \tag{104}$$

For any

$$0 < \eta < \frac{2(1-\delta)}{\alpha}, \tag{105}$$

we have

$$F(\theta_{t+1}) - F_\star \le \left[1 - \frac{2\eta\mu_D}{1+\delta}\left(1 - \frac{\alpha\eta}{2(1-\delta)}\right)\right](F(\theta_t) - F_\star). \tag{106}$$

In particular, choosing

$$\eta = \frac{1-\delta}{\alpha} \tag{107}$$

gives

$$F(\theta_{t+1}) - F_\star \le \left[1 - \frac{\mu_D}{\alpha}\frac{1-\delta}{1+\delta}\right](F(\theta_t) - F_\star). \tag{108}$$

Proof. From $\widehat{D}_t \succeq (1-\delta) D_t$,

$$\frac{|g_{t,i}|}{\hat{d}_{t,i}} \le \frac{|g_{t,i}|}{(1-\delta) d_{t,i}}. \tag{109}$$

Condition equation 103 therefore implies

$$\frac{|g_{t,i}|}{\hat{d}_{t,i}} \le \rho, \tag{110}$$

so clipping is inactive.

Using Theorem 1 with $s_t = \widehat{D}_t^{-1} g_t$,

$$F(\theta_{t+1}) \le F(\theta_t) - \eta\left(1 - \frac{\alpha\eta}{2(1-\delta)}\right) g_t^\top \widehat{D}_t^{-1} g_t. \tag{111}$$

Since

$$\widehat{D}_t \preceq (1+\delta) D_t, \tag{112}$$

inversion reverses the PSD order:

$$\widehat{D}_t^{-1} \succeq \frac{1}{1+\delta} D_t^{-1}. \tag{113}$$

Hence

$$g_t^\top \widehat{D}_t^{-1} g_t \ge \frac{1}{1+\delta} g_t^\top D_t^{-1} g_t \tag{114}$$

$$\ge \frac{2\mu_D}{1+\delta}(F(\theta_t) - F_\star), \tag{115}$$

where the second inequality uses Assumption A4. Substituting equation 115 into equation 111 proves equation 106. Setting $\eta = (1-\delta)/\alpha$ gives equation 108. □

Theorem 2 makes the effect of curvature estimation error explicit. For exact reference curvature, $\delta = 0$, the contraction coefficient is

$$1 - \frac{\mu_D}{\alpha}. \tag{116}$$

With approximate ZO curvature, the effective progress coefficient is reduced continuously by

$$\frac{1-\delta}{1+\delta}. \tag{117}$$

Thus curvature estimation error shrinks the stable stepsize and slows the contraction rate, but does not destroy convergence as long as $\delta < 1$.

H.7 CONVERGENCE OF ZO-CURV

We finally combine the estimator and optimization results over a horizon of T iterations, with one curvature estimate per training step. Throughout, Λ_T, R_T, ϑ_T, ϱ_T, t_0, and δ_T are as in Lemma 4.

Theorem 3 (Convergence of ZO-Curv)**.** Suppose Assumptions A1 (with $R = R_T$), A2, and A3 hold over a horizon of T iterations, and $q \geq \Lambda_T$. Then, with probability at least $1 - \zeta$, simultaneously for every iteration $t_0 \leq t < T$,

$$(1 - \delta_T)D_t \preceq \widehat{D}_t \preceq (1 + \delta_T)D_t. \tag{118}$$

If

$$\delta_T < 1 \tag{119}$$

and

$$\eta \leq \frac{1 - \delta_T}{\alpha}, \tag{120}$$

then

$$\min_{t_0 \leq t < T} \Psi_t \leq \frac{2(F(\theta_{t_0}) - F_\star)}{\eta(T - t_0)}. \tag{121}$$

If, in addition, the iterates $\theta_{t_0}, \ldots, \theta_{T-1}$ remain in the neighborhood $\mathcal{N}$ of Assumption A4 and satisfy the no-clipping condition equation 103 with δ replaced by δ_T, then choosing

$$\eta = \frac{1 - \delta_T}{\alpha} \tag{122}$$

gives

$$F(\theta_T) - F_\star \leq \left[1 - \frac{\mu_D}{\alpha}\frac{1 - \delta_T}{1 + \delta_T}\right]^{T - t_0} (F(\theta_{t_0}) - F_\star). \tag{123}$$

Proof. By Lemma 4, with probability at least $1-\zeta$, $(1-\delta_T)\tilde{v}_t \leq v_t^{\mathrm{ZO}} \leq (1+\delta_T)\tilde{v}_t$ coordinate-wise for every $t_0 \leq t \leq T$. Lemma 5 with $\delta = \delta_T$ then proves equation 118. Equation equation 121 follows from Corollary 1 applied to the iterations $t_0, \ldots, T-1$, and equation 123 follows from Theorem 2 applied at each of these iterations. □

The query budget enters Theorem 3 through the reference preconditioner rather than through δ_T: D_t is built from the damped target $\tilde{h}_t$, whose damping $\lambda_t = \|h_t^{\mathrm{G}}\|_1/q$ decreases as $1/q$, so larger budgets bring the reference update closer to the undamped GNB-preconditioned update of Sophia-G. The estimation error δ_T is controlled by the EMA window and depends on d, q, and T only logarithmically.

Corollary 2 (Controlled penalty relative to reference GNB curvature)**.** Suppose there exists a constant $\bar{\delta} < 1$ such that $\delta_T \leq \bar{\delta}$. Then the number of local iterations $T - t_0$ required to reach

$$F(\theta_T) - F_\star \leq \epsilon_{\mathrm{opt}} \tag{124}$$

satisfies

$$T - t_0 = O\left(\frac{\alpha}{\mu_D}\frac{1 + \bar{\delta}}{1 - \bar{\delta}} \log \frac{F(\theta_{t_0}) - F_\star}{\epsilon_{\mathrm{opt}}}\right). \tag{125}$$

For exact reference curvature, $\bar{\delta} = 0$, this reduces to

$$T_{\mathrm{G}} = O\left(\frac{\alpha}{\mu_D} \log \frac{F(\theta_{t_0}) - F_\star}{\epsilon_{\mathrm{opt}}}\right). \tag{126}$$

Hence the multiplicative iteration penalty of ZO curvature relative to the reference GNB-preconditioned core is bounded by

$$\frac{T_{\mathrm{ZO}}}{T_{\mathrm{G}}} \lesssim \frac{1 + \bar{\delta}}{1 - \bar{\delta}}. \tag{127}$$

For small $\bar{\delta}$,

$$\frac{1 + \bar{\delta}}{1 - \bar{\delta}} = 1 + 2\bar{\delta} + O(\bar{\delta}^2). \tag{128}$$

Proof. From Theorem 3,

$$F(\theta_T) - F_\star \le (1-\nu)^{T-t_0} \left(F(\theta_{t_0}) - F_\star\right), \tag{129}$$

where

$$\nu := \frac{\mu_D}{\alpha} \frac{1-\bar{\delta}}{1+\bar{\delta}}. \tag{130}$$

Using

$$1 - \nu \le e^{-\nu}, \tag{131}$$

it is sufficient that

$$T - t_0 \ge \frac{1}{\nu} \log \frac{F(\theta_{t_0}) - F_\star}{\epsilon_{\text{opt}}}, \tag{132}$$

which gives equation 125. Equation equation 128 is the Taylor expansion around $\bar{\delta} = 0$. □

Remark 2 (Pipelined updates with staleness s)**.** In the deployed pipeline (Section 3.1 and Algorithm 1), the EMA v_t^{ZO} used at step t is built from estimates $h_{t'}^{\mathrm{ZO}}$ queried at earlier snapshots $\theta_{t'}$, $t' \le t - s$, and is held fixed between refreshes. Define the reference EMA $\tilde{v}_t$ with exactly the same schedule, from $\tilde{h}_{t'}$ at the same snapshots, mini-batches, and sampled labels. Proposition 1 is a per-refresh statement. Lemma 4 applies to the sequence of refreshes, with s indexing refreshes and $\mathcal{G}_s$ the history before the directions of refresh s: these directions are still drawn independently of $\mathcal{G}_s$, and a held step copies both EMA states. Lemma 5 depends only on v_t^{ZO} and $\tilde{v}_t$. Theorems 1 and 2 are per-step statements about D_t and $\widehat{D}_t$, with Assumptions A3 and A4 stated relative to the lagged D_t. Theorem 3 unions over at most T refreshes. All results therefore hold verbatim for any staleness s, including the one-step offset of the main experiments and the $s = 3$ deployment of Appendix **??**; staleness enters only through the reference D_t, which the assumptions already constrain.

H.8 Interpretation

The above results establish the following chain:

$$q, \varepsilon \quad \Longrightarrow \quad \tilde{h}_t \text{ (unbiased target)} \quad \Longrightarrow \quad \delta_T \text{ (EMA)} \quad \Longrightarrow \quad \text{optimization rate.} \tag{133}$$

The finite query budget has two separate effects. Its systematic effect is the uniform damping $\lambda_t = \|h_t^{\mathrm{G}}\|_1/q$ of the curvature target (Proposition 1), which averaging before squaring keeps a factor q smaller than squaring before averaging, and which vanishes as $q \to \infty$. Its random effect is a fluctuation of the order of the target in every coordinate, with constants independent of d. The curvature EMA averages this fluctuation over its effective window (Lemma 4), and the positive floor preserves the resulting relative bound (Lemma 5), which yields the multiplicative preconditioner distortion δ_T of equation 60. As long as $\delta_T < 1$, ZO-Curv preserves descent. In the local unclipped regime, the convergence coefficient is reduced from the reference value

$$\frac{\mu_D}{\alpha} \tag{134}$$

to

$$\frac{\mu_D}{\alpha} \frac{1-\delta_T}{1+\delta_T}, \tag{135}$$

where μ_D and α refer to the damped GNB reference preconditioner.

Therefore, curvature-estimation error produces a continuous and controlled degradation of the convergence rate rather than destroying the descent direction. This differs fundamentally from injecting ZO noise directly into the first-order gradient: an erroneous gradient estimate can rotate or reverse the descent direction, whereas a positive curvature estimate only rescales the coordinate-wise first-order update, with clipping additionally bounding the maximum step magnitude.

Thus, replacing the backpropagation-based GNB computation with forward-only ZO queries yields a damped GNB preconditioner: the query budget controls the damping, the curvature EMA controls the estimation error, and the convergence mechanism of clipped curvature preconditioning is retained.