HELENE: Hessian Layer-wise Clipping and Gradient Annealing for Accelerating Fine-Tuning LLM with Zeroth-Order Optimization

Huaqin Zhao[1*] Jiaxi Li[1*] Yi Pan[1] Shizhe Liang[1]
Xiaofeng Yang[2] Fei Dou[1] Tianming Liu[1] Jin Lu[1]
[1]University of Georgia [2]Emory University
Jin.Lu@uga.edu

Abstract

Fine-tuning large language models (LLMs) faces significant memory challenges due to the high cost of back-propagation. MeZO addresses this issue using zeroth-order (ZO) optimization, matching memory usage to inference but suffering from slow convergence due to varying curvatures across model parameters. To overcome this limitation, we propose HELENE, a scalable and memory-efficient optimizer that integrates annealed A-GNB gradients with diagonal Hessian estimation and layer-wise clipping as a second-order pre-conditioner. HELENE provably accelerates and stabilizes convergence by reducing dependence on total parameter space and scaling with the larger layer dimension. Experiments on RoBERTa-large and OPT-1.3B demonstrate superior performances, achieving up to 20× speedup over MeZO with an average accuracy improvement of 1.5%. HELENE also supports full and parameter-efficient fine-tuning methods, outperforming several state-of-the-art optimizers.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across various downstream tasks, and fine-tuning has become the standard approach for adapting LLMs to improve task-specific performance, in which the first-order optimizers such as Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951), Adam (Kingma and Ba, 2014) and AdamW (Loshchilov and Hutter, 2017) are widely used. While effective, however, these methods demand significant memory resources primarily due to the backpropagation, which makes fine-tuning process challenging, especially for large-scale models. To overcome this limitation, Malladi et al. (2023) proposed a memory-efficient zeroth-order optimizer, MeZO, that estimates gradients using only two forward passes per training step, contributing to considerable memory savings.

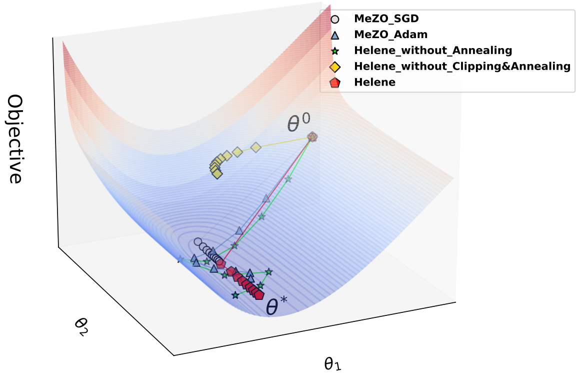


Figure 1: The motivating toy example. HELENE can maintain stable updates when facing curvature issues, while other optimizers are severely affected by them.

However, recent studies show that loss functions in deep learning often exhibit heterogeneous curvatures across different model parameters and layers (Sagun et al., 2016; Ghorbani et al., 2019; Zhang et al., 2022; Yao et al., 2020), which poses challenges to ZO optimization. This variation in curvature can overall hinder training efficiency and lead to sub-optimal results. To address this issue, more advanced techniques are developed, such as incorporating second-order information to better account for curvature differences (Liu et al., 2023; Tran and Cutkosky, 2022; Jahani et al., 2021). However, in ZO optimization, directly computing the Hessian from first-order derivatives is nearly impossible, and partial Hessian evaluations are computationally intensive, leading to slower convergence. Moreover, in practice, we observe that methods estimating second-order information without careful design can fail in fine-tuning LLMs as illustrated in Figure 1.

To overcome the aforementioned challenges, we propose HELENE, a novel optimizer designed to

[*] Equal Contribution

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 26044–26067
November 4-9, 2025 ©2025 Association for Computational Linguistics

estimate second-order curvature information efficiently in the context of ZO optimization. Originating from label-sampling-based Gaussian-Newton -Bartlett (GNB) Estimator (Schraudolph, 2002; Wei et al., 2020), we introduce a label-sampling-free and efficient Hessian estimator called the asymptotic Gauss-Newton-Bartlett (A-GNB) Estimator in our HELENE algorithm, which estimates the diagonal of the Hessian matrix. A-GNB is proven to asymptotically converge to the unbiased Gauss Newton matrix.

Another key innovation of HELENE is its ability to adaptively clip Hessian updates according to the curvature of each layer, which significantly enhances convergence rates. While the convergence of state-of-the-art optimizers like MeZO-Sophia may require $\mathcal{O}(d)$ steps; in contrast, the convergence of HELENE requires significantly less steps, which is $\mathcal{O}(\max_i d_i)$ based on the largest layer dimension $\max_i d_i$ across all layers, making it more suitable for modern deep architectures.

We observed that ZO methods require a stronger emphasis on momentum because of the increasing noise in stochastic approximation (SPSA) gradient estimation as optimization progresses, due to the greater difficulty of training as the model parameters approach a local minimum, causing the noise magnitude from sampling perturbations to exceeding that of the true gradient signal. Therefore, in contrast to most of the momentum design in first-order methods, the proposed HELENE algorithm also integrates a novel annealing exponential moving average (EMA) of the gradients, which dynamically reduces the weight of the gradient in the momentum updates.

Overall, our key contributions can be summarized as follows:

1. HELENE integrates a novel A-GNB estimator that efficiently estimates the diagonal of the Hessian matrix without the need for label sampling which may incur more noise. This estimator asymptotically converges to the unbiased diagonal Gauss Newton matrix, improving the efficiency and precision of curvature-aware updates. We also devise a new layer-wise adaptive clipping mechanism by adjusting Hessian updates according to the curvature of each layer. HELENE integrates a new technique of annealing exponential moving average (EMA) of the gradients, ensuring robustness in non-convex loss landscapes.

2. Our theoretical analysis demonstrates that HELENE achieves improved convergence rates compared to existing methods, particularly for models with many layers. By reducing the convergence steps from $\mathcal{O}(d)$ to $\mathcal{O}(\max_i d_i)$, HELENE is provably more scalable for modern deep learning model architectures, especially for fine-tuning LLMs.

3. HELENE achieves up to 20× speedup compared to MeZO and improves performance by an average of 1.5%. We conduct extensive experiments on RoBERTa-large, OPT-1.3B, and OPT-13B across various downstream tasks to verify HELENE's effectiveness. Furthermore, we demonstrate that HELENE not only remains compatible with both full parameter tuning and PEFT, but also outperforms many of the latest optimizers across a range of tasks.

2 Preliminaries

In this section, we briefly review essential background concepts related to ZO optimization and diagonal Hessian approximation, which are fundamental to the design of our proposed method.

2.1 ZO Gradient Estimators and MeZO

ZO optimization has long been studied in the context of convex and non-convex objectives. One of the typical ZO gradient estimators is the simultaneous perturbation stochastic approximation (SPSA) (Maryak and Chin, 2001). Given a model with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and loss function $\mathcal{L}$, SPSA estimates the gradient on a minibatch $\mathcal{B}$ as:

$$g_\epsilon(\boldsymbol{\theta}) = \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{z}; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \boldsymbol{z}; \mathcal{B})}{2\epsilon} \boldsymbol{z} \quad (1)$$

where $\boldsymbol{z} \in \mathbb{R}^d$ with $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ and ϵ is the perturbation scale.

2.2 Diagonal Hessian Approximation

While MeZO provides valuable tools to estimate a noisy gradient, optimization can be significantly enhanced by incorporating second-order information such as curvature. However, directly computing and applying the full Hessian matrix is computationally expensive, especially in high-dimensional parameter spaces. In particular, applying the Hessian pre-conditioner by computing $\boldsymbol{H}^{-1}\boldsymbol{g}$ at each iteration is notably costly. To address this challenge, inexact Newton methods have been developed, where approximations of the Hessian are used instead of the full matrix (Bollapragada et al., 2019; Xu et al., 2020).

A simple yet effective alternative is to approximate the Hessian by its diagonal elements, which reduces computational complexity while retaining useful curvature information. In this approach, a general descent direction can be written as follows:

$$\Delta\boldsymbol{\theta} \approx \mathrm{diag}(\boldsymbol{H})^{-1}\boldsymbol{g},$$

where $\mathrm{diag}(\boldsymbol{H})$ represents the diagonal elements of the Hessian matrix. This method enables efficient inverse Hessian application and supports inexact Newton methods, which provides improved convergence in complex problems.

3 Method

We first highlight the limitations of existing ZO methods in Section 3.1 to motivate our proposed model. Then, we formally present HELENE in Section 3.2. In Sections 3.3 and 3.4, we introduce Annealing mechanism and A-GNB, followed by a detailed discussion of layer-wise clipped diagonal Hessian in Section 3.5.

3.1 Motivation

Highly variable curvature across different parameters. ZO optimizers, such as MeZO (Malladi et al., 2023), have been proposed to mitigate the substantial GPU memory consumption associated with finetuning, providing memory-efficient solutions by approximating gradients through forward passes. Nevertheless, despite memory savings, existing ZO methods still face challenges in handling the heterogeneous curvatures of LLMs, which can result in inefficient convergence and suboptimal solutions. One key challenge is the inability of optimizers to adapt to the highly variable curvature across different layers and parameters in large models (Sagun et al., 2016; Ghorbani et al., 2019; Zhang et al., 2022). While techniques that estimate second-order information, such as curvature-aware methods, have shown promise in improving optimization efficiency (Liu et al., 2023; Tran and Cutkosky, 2022), they are challenging to integrate into ZO optimizers due to the noise from label-sampling and the difficulty of computing or approximating the Hessian efficiently in high-dimensional spaces.

Limitation of EMA to balance short-term gradient noise and long-term convergence. A commonly used technique to manage these curvature variations is the Exponential Moving Average (EMA), which smooths the gradient updates over iterations. However, EMA alone can be insufficient for highly non-convex loss landscapes, especially when it lacks mechanisms to adaptively adjust the weights between the past momentum and the current gradient in the presence of strong noise in gradient estimation. Without annealing, EMA struggles near stationary points where ZO gradient noise exceeds the true gradient, leading to suboptimal convergence. The noisy gradient estimation issue calls for more careful ZO-specific strategies to balance short-term gradient fluctuations with long-term convergence.

Challenge in managing extreme curvature values using Universal clipping. Furthermore, clipping the Hessian to manage extreme curvature values is another widely adopted strategy. Sophia (Liu et al., 2023), for example, performs global clipping of Hessian-based updates to ensure numerical stability, which essentially can slow down the convergence. While effective at curbing extreme updates, applying a universal clipping threshold across all parameters is inherently suboptimal for models with heterogeneous curvatures. A universal clip might suppress meaningful gradient information in some layers while insufficiently addressing extreme Hessian values in others, thus limiting the optimizer's ability to adaptively handle the diverse learning dynamics across layers (Tran and Cutkosky, 2022).

Algorithm 1 HELENE with Layer-wise Clipping

```
Input: Initial parameters $\boldsymbol{\theta}_1$, step budget $T$, learning rate schedule $\{\eta_t\}_{t=1}^{T}$, hyperparameters $\{\lambda_i\}$ , $\gamma$, $\beta_1$, $\beta_2$, $\epsilon$.
Set $\boldsymbol{m}_0 = 0$, $h_0 = 0$
for $t = 1$ to $T$ do
    Estimate gradient $\boldsymbol{g}_t$ from Eq. (1).
    $\alpha = \mathrm{Anneal}(t)$
    $\boldsymbol{m}_t = \beta_1\boldsymbol{m}_{t-1} + \alpha\boldsymbol{g}_t$
    if $t \bmod k = 1$ then
        Compute diagonal Hessian estimator $\hat{\boldsymbol{h}}_t = \text{A-GNB}(\boldsymbol{\theta}_t)$
        $\boldsymbol{h}_t = \beta_2\boldsymbol{h}_{t-k} + (1-\beta_2)\hat{\boldsymbol{h}}_t$
    else
        $\boldsymbol{h}_t = \boldsymbol{h}_{t-1}$
    end if
    update $\boldsymbol{\theta}_{t+1,i} = \boldsymbol{\theta}_{t,i} - \eta_t \cdot \frac{\boldsymbol{m}_{t,i}}{\gamma\cdot\max(\boldsymbol{h}_{t,i},\lambda_i)+\epsilon}$ for each layer $i$
end for

Subroutine Anneal($t$)
```

$$\alpha \leftarrow \beta_1 + (1-\beta_1)\cdot\exp\left(-t/T\right) \tag{2}$$

To overcome the limitations above, HELENE addresses both the limitations of EMA and the issue of global Hessian clipping. We instantiate MeZO-SGD, MeZO-Adam, and HELENE on a simplified 2D problem to illustrate the advantages of HELENE, as shown in Figure 1. A visual comparison of the methods reveals that while MeZO-Adam and MeZO-SGD struggle to converge effectively, Compared to methods like HELENE without clipping and without annealing, which estimate second-order information but struggle to maintain stability in the presence of heterogeneous curvature, HELENE more effectively captures and handles curvature information.

3.2 HELENE: Hessian Layer-wise Clipping and Gradient Annealing

In HELENE, we introduce an annealing mechanism to mitigate noise dominance in SPSA-estimated gradients, combined with a clipped diagonal Hessian pre-conditioner that adjusts parameter update step sizes based on layer-wise curvature. First, the ZO gradient is estimated using the SPSA, while the diagonal Hessian is efficiently estimated by the proposed new A-GNB method, to eliminate the noise incurred in sampling labels from the model output used in GNB and Sophia. At each iteration, SPSA provides an estimate $\boldsymbol{g}_t$ with random perturbations, and A-GNB returns $\boldsymbol{h}_t$, the diagonal Hessian of the mini-batch loss.

We apply an exponential moving average (EMA) to both the gradient and diagonal Hessian estimates to reduce noise and improve stability. To further enhance convergence, we apply layer-wise magnitude-based clipping to the diagonal Hessian, ensuring extreme values do not disproportionately affect parameter updates. We provide our pseudo code in Algorithm 1 and describe each module in detail in the following section.

3.3 Annealing Mechanism

We observe that, unlike first-order methods such as SGD, ZO methods require stronger momentum due to increasing noise dominance in SPSA gradient estimation as optimization progresses, as illustrated in Appendix Figure 8. **To address this, we introduce an annealing factor that dynamically reduces the reliance on newly sampled gradients.** By gradually diminishing the influence of potentially noisy estimates, the algorithm avoids compounding noise in the Hessian and gradients near optima, leading to more stable second-order updates and faster convergence. The resulting stability and improved convergence behavior can be visualized in Figure 6a in the Appendix. Notably, our annealing approach is simple to implement, requiring the tuning of only a single hyperparameter.

Algorithm 2 Asymptotic Gauss-Newton-Bartlett (A-GNB)

```
Parameters: $\boldsymbol{\theta}$
Draw a mini-batch of the input $\{x_b\}_{b=1}^{B}$
Estimate diagnal Hessian matrix by $\boldsymbol{h} = \sum_{b=1}^{B}[g(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)] \odot [g(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)]$
return $\boldsymbol{h}$
```

At each iteration, the annealing mechanism computes α using an exponential decay schedule in Eq. 2, where T is a predefined hyperparameter controlling the annealing rate. The increase of noise dominance in SPSA is likely due to the greater difficulty of training as the model parameters approach a local minimum, causing the noise magnitude from sampling perturbations to exceed that of the true gradient signal. To address this, as t increases, α gradually decreases to reduce the impact of gradient on the updates, mitigating the bias introduced by EMA. This ensures that, in the later stages of training, the model focuses more on stable gradient estimates and less on noisy or rapidly changing updates via the SPSA estimated gradient. The annealing mechanism is incorporated into the EMA update rule as line 6 in Algorithm 1. Via dynamical α, the annealing mechanism ensures that the optimizer can effectively balance short-term noise with long-term convergence.

3.4 Asymptotic Gauss-Newton-Bartlett (A-GNB) Estimator

The original GNB (Martens, 2020) estimator relies on sampled labels $\hat{y}_b$ drawn from the categorical distribution based on the model's output. However, this induces stochasticity due to label sampling, which could be problematic when label distributions are highly imbalanced, as is the case in LLM training. We propose a new estimator, which we call the **Asymptotic Gauss-Newton-Bartlett (A-GNB) Estimator**, that replaces the sampled labels $\hat{y}_b$ with the true labels y_b and asymptotically converges to the true diagonal of the Gauss-Newton matrix, which is a biased estimator for the diagonal of the Hessian as shown below:

$$\nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \approx J_{\boldsymbol{\theta}} \cdot S \cdot J_{\boldsymbol{\theta}}^{\top} \quad (3)$$

For convenience we denote $J_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, x)$ as $J_{\boldsymbol{\theta}}$ which is the Jacobian of the model's output $f(\boldsymbol{\theta}, \boldsymbol{x})$ with respect to the parameters $\boldsymbol{\theta}$, and $S = \frac{\partial^2 L(t,y)}{\partial t^2}$ is the second-order derivative of the loss w.r.t. the logits $t = f(\boldsymbol{\theta}, \boldsymbol{x})$ and $y \sim p(y|\boldsymbol{x})$, which implies that $S = \mathbb{E}_{y \sim p(\boldsymbol{\theta}, \boldsymbol{x})} \left[\frac{\partial^2 L(t,y)}{\partial t^2} \right]$ assuming that L is the Cross-Entropy loss. Consequently, the diagonal of the Gauss-Newton matrix for the mini-batch loss is estimated as:

$$
\begin{aligned}
& diag(J_{\boldsymbol{\theta}} \cdot S \cdot J_{\boldsymbol{\theta}}^{\top}]) \\
=& \mathbb{E}_{y \sim p(y|\boldsymbol{x})} \left[diag(J_{\boldsymbol{\theta}} \frac{\partial L(t,y)}{\partial t} \frac{\partial L(t,y)}{\partial t}^{\top} J_{\boldsymbol{\theta}} \right] \\
=& \mathbb{E}_{y \sim p(y|\boldsymbol{x})} \left[diag(\nabla_{\boldsymbol{\theta}} L(f(\boldsymbol{\theta}, \boldsymbol{x}), y) \nabla_{\boldsymbol{\theta}} L(f(\boldsymbol{\theta}, \boldsymbol{x}), y)^{\top}) \right] \\
\approx& \frac{1}{B} \sum_{b=1}^{B} [(g(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)] \odot [(g(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)]
\end{aligned}
$$

where $diag(\cdot)$ represents the diagonal elements of a matrix, and B denotes the batch size and g is the estimated gradient from Eq. (1). In contrast to GNB estimator, which includes sampling label $\hat{y}$ from the logit probability output from the model, we replace it by y_b, the true label, thereby avoiding the need for post-output label sampling. By eliminating the stochasticity induced by sampled labels $\hat{y}$, we reduce the variance caused by sampling noise, and it is especially beneficial in imbalanced data scenarios, when samples from minor class is rarely selected unless sampling significantly many times.

The estimated gradient terms now correspond directly to the true labels, and their outer product sums up to the true Gauss-Newton approximation of the Hessian. As the batch size $B \to \infty$, the A-GNB estimator converges to the true Gauss-Newton's diagonal:

$$
\begin{aligned}
& \lim_{B \to \infty} \frac{1}{B} \sum_{b=1}^{B} [g(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)] \odot [g(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)] \\
&= diag(J_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{x}) \cdot S \cdot J_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{x})^{\top})
\end{aligned}
$$

Therefore, The A-GNB estimator asymptotically converges to the true diagonal of the Gauss-Newton matrix as B increases.

3.5 Laywerwise Clipped Diagonal Hessian to help Newton's method

As discussed in the motivating examples, fine-tuning LLMs and optimizing non-convex functions pose challenges for estimating second-order information. Moreover, the inaccuracy of Hessian estimates and changes in the Hessian along the optimization trajectory can render second-order information unreliable. To address these issues, we draw inspiration from Sophia(Liu et al., 2023). While Sophia performs clipping on the Newton update $\boldsymbol{H}^{-1}\boldsymbol{g}$, we propose a more robust approach by applying **layer-wise clipping directly to the Hessian matrix $\boldsymbol{H}$**. Using a universal clipping threshold for the update $\boldsymbol{H}^{-1}\boldsymbol{g}$ disregards the differences in layer-wise Hessian and gradient magnitudes, which are frequently observed during DNN training, and may distort valuable gradient information. Moreover, $\boldsymbol{H}^{-1}\boldsymbol{g}$ introduces excessive bias, potentially distorting useful gradient information, whereas clipping extreme Hessian values more effectively preserves essential second-order information.

In particular, we improve convergence rates by (1) considering only the positive entries of the diagonal Hessian and (2) introducing layer-wise clipping of the Hessian values. This approach adapts the clipping threshold across layers to account for the diverse curvature across different parts of the model. Given a clipping threshold $\lambda_i > 0$ for layer i, the clipping function is defined as:

$$
\mathrm{clip}(\boldsymbol{h}_i) = \max(\boldsymbol{h}_i, \lambda_i), \quad \lambda_i \in \mathbb{R},
$$

where all operations are applied element-wise for each layer. The update rule for layer i is then written as:

$$
\boldsymbol{\theta}_{t+1,i} = \boldsymbol{\theta}_{t,i} - \eta \cdot \frac{\boldsymbol{m}_{t,i}}{\gamma \cdot \max(\boldsymbol{h}_{t,i}, \lambda_i) + \epsilon},
$$

where $\epsilon > 0$ is a small constant to avoid division by zero, and λ_i controls the fraction of clipped Hessian values per layer, more layer-wise-clipping visualization and numerical experiments can be found in E.2.

For further information about the differences of HELENE with previous work, please refer to the related work section in Appendix A.

4 Convergence Analysis

In this section, we provide a theoretical analysis of the convergence of our proposed method.

The theoretical bound for the number of steps T in our method is given by the following theorem with two assumptions:

Assumption 1. *Let $L : \mathbb{R}^d \to \mathbb{R}$ be a loss function. We assume L is twice continuously differentiable strictly convex, and has a unique minimizer denoted by $\boldsymbol{\theta}^*$. For each layer i, define μ_i as the minimum eigenvalue of the Hessian matrix of L*

concerning the parameters of that layer evaluated at its minimizer:

$$\mu_i \equiv \lambda_{\min}(\nabla^2_{\boldsymbol{\theta}_i} L(\boldsymbol{\theta}^*))$$

where $\nabla^2_{\boldsymbol{\theta}_i}$ denotes the Hessian with respect to the parameters of the i-th layer.

Assumption 2. *Regarding the Hessian $\nabla^2 L(\boldsymbol{\theta})$ of the loss function, we assume: There exists a radius $R_i > 0$ such that for any $\boldsymbol{\theta}_i, \boldsymbol{\theta}'_i \in \mathbb{R}^d$ with $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}'_i\|_2 \leq R_i$, the following inequality holds:*

$$\left\|\nabla^2 L(\boldsymbol{\theta}'_i \mid \boldsymbol{\theta}_{-i})^{-1} \nabla^2 L(\boldsymbol{\theta}_i \mid \boldsymbol{\theta}_{-i})\right\|_2 \leq 2$$

where $\|\cdot\|_2$ represents the spectral norm.

Theorem 1. *Under Assumptions 1 and 2, let $\eta = \frac{1}{2}$ and $\lambda_i = \frac{R_i}{2\sqrt{d_i}}$. The update reaches a loss at most ϵ in*

$$T \leq \max_i \left\{ d_i \cdot (L(\boldsymbol{\theta}_{0,i}) - \min L) + \ln\left(\frac{\mu_i R_i^2}{32 d_i \epsilon}\right)\right\}$$

steps, where L is the loss function, $\boldsymbol{\theta}_{0,i}$ is the initial parameter vector for layer i, μ_i is the strong convexity constant for layer i, and R_i is the bound on the distance between $\boldsymbol{\theta}_{0,i}$ and $\boldsymbol{\theta}^*_i$.

The best known theoretical bound for the number of steps T to reach a loss at most ϵ when using noisy gradient estimation is given by Sophia in which $T_{\text{SOPHIA}} \sim \mathcal{O}(d)$, where d is the total dimension of the parameter space. This result implies that the convergence rate depends linearly on the total dimension d, which can lead to slow convergence for models with large parameter spaces. In contrast, our method introduces layer-wise parameters $\lambda_i = \frac{R_i}{2\sqrt{d_i}}$, where R_i is the bound on the distance between the initial parameters $\boldsymbol{\theta}_{0,i}$ and the optimal parameters $\boldsymbol{\theta}^*_i$ for layer i, and d_i is the dimension of the parameter space for layer i. This layer-wise setting significantly reduces the complexity to $T_{\text{SOPHIA}} \sim \mathcal{O}(\max_i d_i)$, which is the maximum dimension across layers. Besides the lower runtime bound, our method allow each layer to have its own parameter ρ_i, allowing the method to adapt to the specific geometry of each layer. Refer to Appendix E.3 for the empirical study on the significant variance using unified parameter clipping across different layers. This flexibility leads to a more efficient optimization process, as each layer is treated independently based on its characteristics. In large models where some layers have much smaller dimensions than others, our method is able to achieve faster convergence by focusing on the most difficult layer with the largest dimension, therefore making our method more scalable for deep models with many layers. Detailed proof can be seen in the Appendix F.

5 Experiments

To rigorously evaluate the capability and universality of HELENE, we follow the experimental settings in MeZO on both medium-sized masked LMs (RoBERTa-large (Liu, 2019), 350M) and auto-regressive LLMs (OPT-1.3B, OPT-13B (Zhang et al., 2023), and LLaMA-2-7B (Touvron et al., 2023)) under both few-shot and many-shot settings. Additionally, all optimization algorithms are tested with full-parameter fine-tuning (FT) and two PEFT methods including LoRA (Hu et al., 2021) and prefix-tuning (Li and Liang, 2021). We also conduct experiments with zeroth-order (ZO) versions of selected optimizers, as well as ZO-SGD variants introduced in (Zhang et al., 2024). In addition, we perform a comprehensive analysis of wall-clock time and hyperparameter sensitivity to assess the practical efficiency and robustness of each method.

5.1 Masked Language Models

For masked LMs, we conduct experiments using RoBERTa-large on three types of NLP tasks, sentiment classification, natural language inference, and topic classification with $k = 16$ examples per class. We run HELENE for 5,000 steps and FT for 1,000 steps. We include baseline methods of MeZO (Malladi et al., 2023), ZO-AdaMU (Jiang et al., 2024) and MeZO-SVRG (Gautam et al., 2024). The experimental results are listed in Table 1, from which we can have following findings.

HELENE largely outperforms zero-shot and linear probing. Across all six datasets, HELENE can stably optimize the pre-trained LM and consistently perform better than zero-shot and linear probing.

Significant speed improvement over MeZO. With the guidance of layer-wise clipped Hessian information, HELENE can reach convergence in about 5000 steps on average across the datasets, accelerating the optimization process by approximate **10**× than MeZO, based on the average results. Meanwhile, the results show that HELENE can still achieve performance on par with MeZO, leading in 5 out of 6 datasets in terms of average accuracy across three tuning methods.

Task	**SST-2**	**SST-5**	**SNLI**	**MNLI**	**RTE**	**TREC**
Task Type	—— sentiment ——		—— natural language inference ——			— topic —
Zero-shot	79.0	35.5	50.2	48.8	51.4	32.0
LP	76.0 (±2.8)	40.3 (±1.9)	66.0 (±2.7)	56.5 (±2.5)	59.4 (±5.3)	51.3 (±5.5)
FT	91.9 (±1.8)	46.7 (±1.9)	77.5 (±2.6)	70.0 (±2.3)	66.4 (±7.2)	85.0 (±2.5)
FT(LoRA)	91.4 (±1.7)	46.7 (±1.1)	74.9 (±4.3)	67.7 (±1.4)	66.1 (±3.5)	82.7 (±4.1)
FT(Prefix)	91.9 (±1.0)	47.7 (±1.1)	77.2 (±1.3)	66.5 (±2.5)	66.6 (±2.0)	85.7 (±1.3)
MeZO	90.5 (±1.2)	45.5 (±2.0)	68.5 (±3.9)	58.7 (±2.5)	64.0 (±3.3)	76.9 (±2.7)
MeZO (LoRA)	91.4 (±0.9)	43.0 (±1.6)	69.7 (±6.0)	64.0 (±2.5)	64.9 (±3.6)	73.1 (±6.5)
MeZO (Prefix)	90.8 (±1.7)	45.8 (±2.0)	71.6 (±2.5)	63.4 (±1.8)	65.4 (±3.9)	**80.3** (±3.6)
ZO-AdaMU	92.6 (±0.4)	45.6 (±0.4)	71 (±1.3)	63.8 (±0.6)	66.0 (±2.3)	77.1 (±1.7)
MeZO-SVRG	91.7 (±0.2)	46.0 (±0.5)	67 (±1.5)	43 (±0.9)	65.7 (±1.3)	74 (±1.2)
HELENE	**92.6** (±2.3)	**46.7** (±0.8)	**72.0** (±2.6)	58.9 (±1.1)	65.7 (±1.2)	78.1 (±1.5)
HELENE-F	91.8 (±1.7)	44.8(±1.1)	70.3(±1.5)	58.7 (±0.6)	63.8 (±0.9)	77.5 (±3.1)
HELENE (LoRA)	90.6 (±0.3)	41.8 (±1.0)	68.5 (±2.0)	59.0 (±1.1)	**66.8** (±3.2)	67.4 (±2.1)
HELENE (Prefix)	91.7 (±0.6)	46.0 (±0.7)	69.5 (±1.9)	**64.6** (±2.1)	66.1 (±1.8)	77.4 (±2.1)

Table 1: Experiments on RoBERTa-large (350M parameters, $k = 16$). HELENE-F compresses the momentum and hessian in the optimizer through matrix factorization. All reported numbers are averaged accuracy (standard deviation) across 5 runs.

Model	Method	**SST-2**	**RTE**	**CB**	**WSC**	**WIC**	**COPA**	**SQuAD**	**Avg.**
LLaMA2-7B	MeZO	92.0	69.5	64.3	62.9	57.8	84.0	72.5	71.9
	HELENE	**93.2**	**71.0**	**71.1**	**63.4**	**59.0**	**86.4**	**77.0**	**74.4**
	HELENE-F	92.4	69.8	69.8	63.2	58.4	86.3	76.0	73.7
OPT-13B	MeZO	91.4	66.1	66.0	62.5	59.4	88.0	84.7	74.0
	HELENE	**92.8**	**68.5**	**69.6**	**64.0**	**61.1**	88.8	82.5	**75.3**
	HELENE-F	92.3	67.2	69.5	63.6	59.6	**88.9**	**85.0**	75.2

Table 2: Experiments on LLaMA-2-7B and OPT-13B using 1,000 examples, highlighting the best-performing results.

5.2 Auto-regressive LLMs

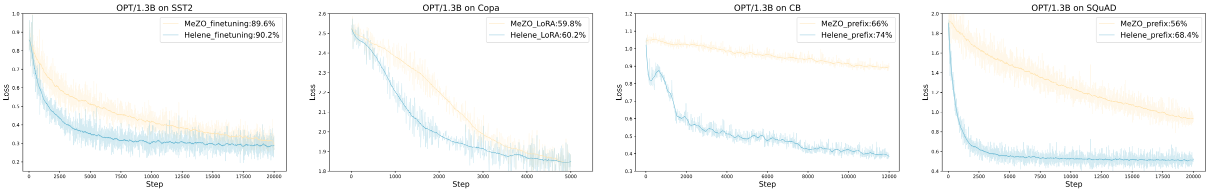
We extend our evaluation to the auto-regressive LLMs OPT-1.3B, OPT-13B, and LLaMA-2-7B across three representative tasks: text classification, multiple choice, and text generation. We employ datasets from the SuperGLUE benchmark (Wang et al., 2019), including SST-2, RTE, CB, WSC, WIC, COPA, and ReCoRD, and additionally evaluate on BoolQ (Clark et al., 2019) and SQuAD (Rajpurkar, 2016). For each dataset, we fine-tune HELENE for approximately 20,000 training steps. Detailed results for OPT-1.3B can be found in the Table 9. The main results are summarized in Table 2, from which we draw the following observations.

HELENE consistently outperforms MeZO across models and tasks. As shown in Table 2, HELENE demonstrates superior performance compared to MeZO on both LLaMA-2-7B and OPT-13B, highlighting its effectiveness and robustness across various language understanding and reasoning tasks. On LLaMA-2-7B, HELENE improves over MeZO by 6.8% on CB, 4.5% on SQuAD, and 2.4% on COPA. Similarly, for OPT-13B, HELENE yields gains of 3.6% on CB and 0.6% on COPA, while maintaining competitive performance on other tasks.

Acceleration and compatibility with PEFT methods. In Figure 2, we visualize the results from four selected datasets across different tasks under three tuning methods. The results indicate that HELENE consistently accelerates convergence by up to 20× while also improving performance. We also conducted experiments to OPT-13B model and evaluating its performance on non-differentiable optimization objectives (F1). For detailed results, please refer to Appendix D.

5.3 Experiments with Other ZO Algorithms

It is worth noting that the ZO optimization technique utilized in Malladi et al. (2023) is primarily the basic SGD version (ZO-SGD), and it is still not clear how effective HELENE is when comparing with other ZO optimization algorithms like ZO-SGD, ZO-SGD-MMT, ZO-SGD-Cons, ZO-SGD-Sign and ZO-Adam as introduced in Liu et al. (2020). Therefore, we reference the statistics of performances summarized in Zhang et al. (2024) and experiment under the same setting with them (Table 8). We further implement the ZO versions



(a) OPT-1.3B on SST2 (b) OPT-1.3B on COPA (c) OPT-1.3B on CB (d) OPT1.3B on SQuAD

Figure 2: Performance and convergence of MeZO and HELENE for fine-tuning, LoRA, and prefix-tuning of OPT-1.3B on different datasets. HELENE achieves approximate $10\times$ speedup and up to 15% accuracy improvement compared to MeZO.

of Adam, AdamW and Lion (Chen et al., 2024) and plot the results in Figure 3. The results indicate that HELENE helps the model converge faster as well as obtain lower validation loss value.

Model	Method	Steps to Convergence	Time to Convergence	Accuracy	Speedup over MeZO
RoBERTa-Large	MeZO	200,000	400 minutes	90.5	1×
RoBERTa-Large	HELENE	5,000	20 minutes	92.6	20×
OPT-1.3B	MeZO	20,000	95 minutes	89.6	1×
OPT-1.3B	HELENE	6,000	45 minutes	91.2	2.11×
LLaMA-7B	MeZO	35,000	300 minutes	92.0	1×
LLaMA-7B	HELENE	10,000	80 minutes	93.2	3.75×

Table 3: Comparison of MeZO and HELENE in terms of convergence steps, time, accuracy, and speedup across different models.

5.4 Memory Usage and Wall-Clock Time Analysis

As shown in Figure 9, HELENE increases the memory usage compared to MeZO because of the storage of the momentum and the diagonal Hessian (details in Table 12). To further reduce memory consumption, we propose HELENE-F, the low-rank implementation of HELENE, inspired by Adafactor (Shazeer and Stern, 2018) and SMMF(Park and Lee, 2024). The detailed algorithm can be seen in Appendix 3. As a result, HELENE-F increases less than 25% memory compared to MeZO, while maintaining the original performance of HELENE, as shown in Figure C. HELENE-F is, to the best of our knowledge, the first ZO algorithm that integrates factorization-based techniques to enhance memory efficiency via compression with both curvature and momentum awareness. Empirical results imply that HELENE-F ensures essential model update dynamics are retained post-decompression, as demonstrated in Figure 4. In Table 3, we present a wall-clock time analysis alongside comparisons of convergence speed, final accuracy, and training efficiency across different model architectures, including RoBERTa-Large, OPT-1.3B, and LLaMA-7B. The consistent speedups observed across these diverse backbones demonstrate the compatibility and adaptability of our approach to various model families.

5.5 Hyperparameter Sensitivity Analysis

We set `beta1` and `beta2` following the default configuration of the Adam (Kingma and Ba, 2014) optimizer. In our method, `T` denotes the total training step budget, while `f` represents the Hessian update frequency. As shown in Table 4, the performance remains stable across different settings of `T` and `f`, demonstrating the robustness of our approach to these hyperparameter choices.

OPT-1.3B on SST-2	T=4000	T=6000	T=8000
Freq = 1	91.1%	90.8%	90.7%
Freq = 5	91.2%	91.0%	91.0%
Freq = 10	90.8%	90.5%	90.6%

Table 4: Accuracy (%) of OPT-1.3B fine-tuned on SST-2 with different step budgets and Hessian update frequency.

5.6 Ablation Study

We conduct a comprehensive ablation study on the key techniques of HELENE in Appendix E, including in-depth analysis of the effects of magnitude clipping across different ranges. Additionally, we explore the factors resulting in initial convergence and subsequent divergence when our proposed modules are not deployed.

6 Conclusion

We introduce HELENE, a novel optimizer for fine-tuning LLMs. It features an asymptotic Gauss-Newton-Bartlett (A-GNB) estimator for unbiased Hessian approximation and a layer-wise clipping mechanism for adaptive updates. HELENE improves stability, scalability, and convergence speed. Experiments on RoBERTa-large, OPT-1.3B/13B, and LLaMA-2-7B demonstrate remarkable speedup over MeZO, comparable memory consumption, and accuracy gains. Compatible with both full and parameter-efficient fine-tuning, HELENE consistently outperforms state-of-the-art optimizers across model scales.

Limitations

While layer-wise clipping alleviates some issues caused by large curvature differences, the diagonal Hessian itself may still be a coarse approximation when the Hessian's off-diagonal terms are significant (e.g., in strongly coupled parameter updates). Our future work will incorporate block-diagonal or structured approximations within a ZO framework.

References

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

Jimmy Ba, Roger B Grosse, and James Martens. 2017. Distributed second-order optimization using kronecker-factored approximations. In *ICLR (Poster)*.

S BECKER. 1988. Improving the convergence of back-propagation learning with second order method. In *Proceedings of the 1988 Connectionist Models Summer School, San Mateo, CA*. Morgan Kaufmann.

Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. 2019. Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578.

Charles George Broyden. 1970. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90.

HanQin Cai, Yuchen Lou, Daniel McKenzie, and Wotao Yin. 2021. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. In *International Conference on Machine Learning*, pages 1193–1203. PMLR.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. 2024. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36.

Xiangyi Chen, Steven Z Wu, and Mingyi Hong. 2020. Understanding gradient clipping in private sgd: A geometric perspective. *Advances in Neural Information Processing Systems*, 33:13773–13782.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. 2000. *Trust-Region Methods*. Society for Industrial and Applied Mathematics, USA.

Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. 2024. Variance-reduced zeroth-order methods for fine-tuning language models. *arXiv preprint arXiv:2404.08080*.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. 2018. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in Neural Information Processing Systems*, 31.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. 2019. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR.

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

Davood Hajinezhad and Michael M Zavlanos. 2018. Gradient-free multi-agent nonconvex nonsmooth optimization. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4939–4944. IEEE.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Majid Jahani, Sergey Rusakov, Zheng Shi, Peter Richtárik, Michael W Mahoney, and Martin Takáč. 2021. Doubly adaptive scaled algorithm for machine learning using second-order information. *arXiv preprint arXiv:2109.05198*.

Shuoran Jiang, Qingcai Chen, Youcheng Pan, Yang Xiang, Yukang Lin, Xiangping Wu, Chuanyi Liu, and Xiaobao Song. 2024. 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, pages 18363–18371.

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

Jiaxi Li, Kiran Davuluri, Khairul Mottakin, Zheng Song, Fei Dou, and Jin Lu. 2024. Proximal federated learning for body mass index monitoring using commodity wifi. In *Proceedings of the 30th Annual International*

Conference on Mobile Computing and Networking, pages 2061–2065.

Jiaxi Li, Yiwei Wang, Kai Zhang, Yujun Cai, Bryan Hooi, Nanyun Peng, Kai-Wei Chang, and Jin Lu. 2025. Fact or guesswork? evaluating large language model's medical knowledge with structured one-hop judgment. *arXiv preprint arXiv:2502.14275*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. 2023. Sophia: A scalable stochastic second-order optimizer for language model pretraining. *arXiv preprint arXiv:2305.14342*.

Mingrui Liu, Zhenxun Zhuang, Yunwen Lei, and Chunyang Liao. 2022. A communication-efficient distributed gradient clipping algorithm for training deep neural networks. *Advances in Neural Information Processing Systems*, 35:26204–26217.

Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. 2020. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54.

Wenqi Liu, Xuemeng Song, Jiaxi Li, Yinwei Wei, Na Zheng, Jianhua Yin, and Liqiang Nie. 2025. Mitigating hallucination through theory-consistent symmetric multimodal preference optimization. *arXiv preprint arXiv:2506.11712*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

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

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

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

James Martens et al. 2010. Deep learning via hessian-free optimization. In *Icml*, volume 27, pages 735–742.

John L Maryak and Daniel C Chin. 2001. Global random optimization by simultaneous perturbation stochastic approximation. In *Proceedings of the 2001 American control conference.(Cat. No. 01CH37148)*, volume 2, pages 756–762. IEEE.

Yurii Nesterov and Boris T Polyak. 2006. Cubic regularization of newton method and its global performance. *Mathematical programming*, 108(1):177–205.

Kwangryeol Park and Seulki Lee. 2024. Smmf: Square-matricized momentum factorization for memory-efficient optimization. *arXiv preprint arXiv:2412.08894*.

R Pascanu. 2013. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*.

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Levent Sagun, Leon Bottou, and Yann LeCun. 2016. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*.

Tom Schaul, Sixin Zhang, and Yann LeCun. 2013. No more pesky learning rates. In *International conference on machine learning*, pages 343–351. PMLR.

Nicol N Schraudolph. 2002. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

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

Yujie Tang, Junshan Zhang, and Na Li. 2020. Distributed zero-order algorithms for nonconvex multiagent optimization. *IEEE Transactions on Control of Network Systems*, 8(1):269–281.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hoang Tran and Ashok Cutkosky. 2022. Better sgd using second-order momentum. *Advances in Neural Information Processing Systems*, 35:3530–3541.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Colin Wei, Sham Kakade, and Tengyu Ma. 2020. The implicit and explicit regularization effects of dropout. In *International conference on machine learning*, pages 10181–10192. PMLR.

Peng Xu, Fred Roosta, and Michael W Mahoney. 2020. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 184(1):35–70.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. 2020. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE.

Haishan Ye, Zhichao Huang, Cong Fang, Chris Junchi Li, and Tong Zhang. 2018. Hessian-aware zeroth-order optimization for black-box adversarial attack. *arXiv preprint arXiv:1812.11377*.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.

Lin Zhang, Shaohuai Shi, and Bo Li. 2022. Eva: Practical second-order optimization with kronecker-vectorized approximation. In *The Eleventh International Conference on Learning Representations*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2023. Opt: Open pre-trained transformer language models, 2022. *URL https://arxiv. org/abs/2205.01068*, 3:19–0.

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

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor W Tsang. 2024. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. *arXiv preprint arXiv:2402.15173*.

Shuang Zhou, Wenya Xie, Jiaxi Li, Zaifu Zhan, Meijia Song, Han Yang, Cheyenna Espinoza, Lindsay Welton, Xinnie Mai, Yanwei Jin, et al. 2025. Automating expert-level medical reasoning evaluation of large language models. *arXiv preprint arXiv:2507.07988*.

A Related work

A.1 Zero-order Optimization

Zeroth-order (ZO) optimization, which only relies on the forward passes of neural networks, offers significant memory savings during the training process. Recently, MeZO (Malladi et al., 2023) adapted the traditional zeroth-order SGD optimization method for fine-tuning LMs, achieving performance comparable to full-parameter fine-tuning while significantly reducing memory usage. Thus, zeroth-order optimization is regarded as a promising approach for memory-efficient fine-tuning of LLMs. Several studies have aimed to improve the MeZO algorithm. For instance, Gautam et al. (2024) introduced a zeroth-order optimization algorithm that integrates both full-batch and mini-batch information to produce asymptotically unbiased, low-variance gradient estimations. However, the convergence rate of their approach still leaves room for improvement. In pursuit of better gradient estimation, Jiang et al. (2024) proposed an innovative perturbation sampling technique inspired by the Adam optimizer. Other methods, such as SPSA (Spall, 1992; Maryak and Chin, 2001), have shown to be effective in non-convex multi-agent optimization (Tang et al., 2020; Hajinezhad and Zavlanos, 2018) and in generating black-box adversarial examples (Chen et al., 2017; Cai et al., 2021; Ye et al., 2018; Li et al., 2025).

A.2 Second-order Information for Fine-tuning LLMs

Classic second-order optimization algorithms precondition the gradient with curvature information (Broyden, 1970; Nesterov and Polyak, 2006; Conn et al., 2000). Over the years, people have developed numerous ways to adapt these methods to deep learning. To the best of our knowledge, BECKER (1988) was the first to use diagonal Hessian as the pre-conditioner. (Martens et al., 2010) approximated the Hessian with conjugate gradient. Schaul et al. (2013) automatically tuned the learning rate of SGD by considering diagonal Hessian. Pascanu (2013) considered Gaussian Newton's approximation of Hessian and Fisher information matrix. Martens and Grosse (2015) and follow-up works (Ba et al., 2017; George et al., 2018; Zhang et al., 2022; Li et al., 2024; Zhou et al., 2025) proposed to approximate the Hessian based on the structure of neural networks. Despite these progress on deep learning applications, for decoder-only LLMs, Adam still appears to be the most popular optimizer. The authors of this paper suspect that many previous second-order optimizers face the challenge that the computational / memory overhead due to frequent Hessian computation hinders improvements in wall-clock time (Gupta et al., 2018). Some of them also depend on specific model architecture or hardware structures, e.g., Anil et al. (2020) offloads hessian computation to CPUs, and George et al. (2018) needs ResNet and very large batch size to approximate the Fisher information matrix. To the best of our knowledge, there was no previous report that second-order optimizers can achieve a speed-up on LLMs in total compute.

There is also a concurrent work HiZOO (Zhao et al., 2024) that utilizes Hessian information to enhance zeroth-order optimization for fine-tuning LLMs. A major focus of HiZOO is to introduce one more forward pass to handle heterogeneous curvatures across parameter dimensions. However, our work focus on incorporating layer-wise clipping to exclude extreme Hessian values and Exponential Moving Average (EMA) to improve generalization.

A.3 Gradient Clipping

Global gradient clipping has been a widely adopted practice in fine-tuning LLMs (Chen et al., 2020; Zhang et al., 2019; Liu et al., 2022, 2025). This technique stabilizes training by mitigating the impact of rare examples and large gradient noise. In addition to gradient clipping, HELENE is the first method to clip the Hessian matrix in second-order optimization techniques. This approach addresses the issue of the Hessian matrix fluctuating along the optimization trajectory and reduces the errors in Hessian approximations.

B Additional Experiments

B.1 ZO Compatibility with LoRA

As shown in Table 5, full tuning with a first-order optimizer like SGD consumes a large amount of memory (27.0 GB), which is substantially reduced by applying LoRA (10.4 GB). Interestingly, our zeroth-order method HELENE-F already uses much less memory in full tuning (5.5 GB), and when combined with LoRA, it achieves the lowest memory footprint (4.4 GB). This illustrates an intuitive and consistent trend: zeroth-order training not only avoids gradient storage but also scales particularly well with LoRA-based fine-tuning.

Method	Model	Full Tuning (FT)	LoRA
SGD (First-order)	OPT-1.3B	27.0 GB	10.4 GB
HELENE-F (ZO)	OPT-1.3B	**5.5 GB**	**4.4 GB**

Table 5: (b) Comparison with first-order LoRA: HELENE-F achieves significantly lower memory usage in both full tuning and LoRA settings.

B.2 Regarding Batch Size and Memory Efficiency

It is true that zeroth-order (ZO) methods benefit from larger batch sizes to reduce optimization noise, increasing the batch size does not lead to significantly higher memory usage. This is because ZO methods avoid the backward pass and do not require activation caching during training.

Table 6 shows the memory usage when finetuning OPT-1.3B on the SST-2 dataset. As shown, first-order (FO) methods require substantially more memory, especially as the batch size increases (e.g., 22.5 GB for batch size 4 and 27.0 GB for batch size 16). In contrast, ZO methods maintain a constant and minimal memory footprint of approximately 4.4 GB, even when the batch size scales up to 64. This is due to the fact that, in PyTorch, ZO finetuning does not set `requires_grad=True`, and therefore no intermediate activations are retained. As a result, memory usage remains nearly flat across different batch sizes.

Finetuning Type	Batch Size	Memory Usage (GB)
First-order (FO)	4	22.5
First-order (FO)	16	27.0
Zeroth-order (ZO)	8 / 16 / 32 / 64	4.4

Table 6: Memory usage when fine-tuning OPT-1.3B on SST-2 with different batch sizes.

B.3 Regarding the Memory Usage of HELENE-F and Contemporary ZO Methods

Recent zeroth-order (ZO) optimizers such as MeZO-SVRG (Gautam et al., 2024) and ZO-AdaMU (Jiang et al., 2024) require nearly twice the memory of MeZO due to their use of additional gradient accumulators or control variates.

As shown in Table 7, HELENE-F consistently performs the best or matches the strongest baseline across all three tasks. This demonstrates that HELENE-F achieves a favorable trade-off between memory efficiency and convergence performance, making it a practical choice for resource-constrained scenarios.

Model	SST-2 (%)	SST-5 (%)	RTE (%)	Memory (GB)
HELENE-F	**92.6**	**46.7**	**66.8**	**5.5**
ZO-AdaMU	**92.6**	45.6	66.0	8.8
MeZO-SVRG	91.7	46.0	65.7	8.8

Table 7: Comparison of recent ZO optimizers in terms of accuracy and memory usage (OPT-1.3B, SST-2/SST-5/RTE).

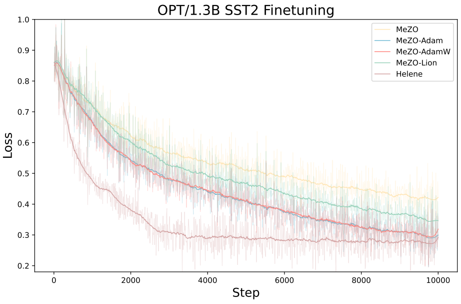


Figure 3: Validation losses for ZO-optimizers. MeZO:0.426, Adam:0.286, AdamW:0.351, Lion:0.343, HELENE:0.283.

C HELENE Variants

C.1 HELENE-F

Due to the necessity of storing Hessian information, HELENE incurs additional memory overhead associated with maintaining both momentum and Hessian matrices. Specifically, each of these components requires storage equivalent to the size of the model parameters. To mitigate this limitation, we propose HELENE-F, a low-rank storage approach for momentum and Hessian, inspired by Adafactor(Shazeer and Stern, 2018) and SMMF (Park and Lee, 2024). HELENE-F leverages factorization-based techniques to enhance memory efficiency through compression while preserving both curvature and momentum awareness. Extensive experiments are conducted to evaluate the effectiveness of HELENE-F, with the results presented in Figure 4.

SST2	Roberta-Large			OPT-1.3B		
	FT	LoRA	Prefix	FT	LoRA	Prefix
FO-SGD	91.4	91.2	89.6	91.1	93.6	93.1
Forward-Grad	90.1	89.7	89.5	90.3	90.3	90.0
ZO-SGD	89.4	90.8	90.0	90.8	90.1	91.4
ZO-SGD-MMT	89.6	90.9	90.1	85.2	91.3	91.2
ZO-SGD-Cons	89.6	**91.6**	90.1	88.3	90.5	81.8
ZO-SGD-Sign	52.5	90.2	53.6	87.2	91.5	89.5
ZO-Adam	89.8	89.5	90.2	84.4	**92.3**	91.4
HELENE	**92.6**	90.6	**91.7**	**90.8**	91.4	**92.4**
HELENE-F	91.8	91.3	91.2	91.0	91.2	90.9

Table 8: Performance of LLM fine-tuning on SST2 over pre-trained Roberta-Large and OPT-1.3B. Best performance among ZO methods (including Forward-Grad) are in **bold**.

Algorithm 3 HELENE-F

```
**Input:** Initial parameters $\boldsymbol{\theta}_1$, total steps $T$, learning rate schedule $\{\eta_t\}_{t=1}^{T}$, hyperparameters $\{\lambda_i\}$, $\gamma$, $\beta_1$, $\beta_2$, $\epsilon$.
Compression/Decompression routines: Decompression($\cdot$), Compression($\cdot$).
Initial compressed momenta $(\mathbf{r}_{\mathbf{m}_0}, \mathbf{c}_{\mathbf{m}_0}, \mathbf{S}_{\mathbf{m}_0})$ and $(\mathbf{r}_{\mathbf{h}_0}, \mathbf{c}_{\mathbf{h}_0}, \mathbf{S}_{\mathbf{h}_0})$.
**for** $t = 1$ **to** $T$ **do**
    $\boldsymbol{g}_t \leftarrow \nabla L_t(\boldsymbol{\theta}_t)$
    $\alpha \leftarrow$ Anneal($t$)
    $\hat{\mathbf{m}}_{t-1} \leftarrow$ Decompression$(\mathbf{r}_{\mathbf{m}_{t-1}}, \mathbf{c}_{\mathbf{m}_{t-1}}, \mathbf{S}_{\mathbf{m}_{t-1}})$
    $\hat{\mathbf{h}}_{t-1} \leftarrow$ Decompression$(\mathbf{r}_{\mathbf{h}_{t-1}}, \mathbf{c}_{\mathbf{h}_{t-1}}, \mathbf{S}_{\mathbf{h}_{t-1}})$
    $\hat{\mathbf{m}}_t \leftarrow \beta_1 \hat{\mathbf{m}}_{t-1} + \alpha\, \boldsymbol{g}_t$
    **if** $t \bmod k = 1$ **then**
        $\hat{\hat{\mathbf{h}}}_t \leftarrow$ A-GNB$(\boldsymbol{\theta}_t)$
        $\hat{\mathbf{h}}_t \leftarrow \beta_2\, \hat{\mathbf{h}}_{t-k} + (1 - \beta_2)\, \hat{\hat{\mathbf{h}}}_t$
    **else**
        $\hat{\mathbf{h}}_t \leftarrow \hat{\mathbf{h}}_{t-1}$
    **end if**
    $(\mathbf{r}_{\mathbf{m}_t}, \mathbf{c}_{\mathbf{m}_t}, \mathbf{S}_{\mathbf{m}_t}) \leftarrow$ Compression$(\hat{\mathbf{m}}_t)$
    $(\mathbf{r}_{\mathbf{h}_t}, \mathbf{c}_{\mathbf{h}_t}, \mathbf{S}_{\mathbf{h}_t}) \leftarrow$ Compression$(\hat{\mathbf{h}}_t)$
    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_t - \eta_t\, \epsilon\, \boldsymbol{\theta}_t$
    **for** each layer $i$ **do**
        $\boldsymbol{\theta}_{t+1,i} \leftarrow \boldsymbol{\theta}_{t,i} - \eta_t \cdot \frac{\hat{\mathbf{m}}_{t,i}}{\gamma\ \max(\hat{\mathbf{h}}_{t,i}, \lambda_i) + \epsilon}$
    **end for**
**end for**
**Subroutine** Anneal($t$):
    $\alpha \leftarrow \beta_1 + (1 - \beta_1) \cdot \exp(-t/T)$
```

Algorithm 4 Decompression

Input: Factorized vectors $\mathbf{r} \in \mathbb{R}^{\hat{n}\times 1}$ and $\mathbf{c} \in \mathbb{R}^{1\times \hat{m}}$, and a binary sign matrix $\mathbf{S} \in \{0,1\}^{\hat{n}\times\hat{m}}$.
Output: Decompressed matrix $\mathbf{M} \in \mathbb{R}^{\hat{n}\times\hat{m}}$ $\mathbf{M} = \mathbf{r} \otimes \mathbf{c}$
$\{\otimes$ is the outer product$\}$

$$M_{i,j} = \begin{cases} M_{i,j}, & \text{if } S_{i,j} = 1 \\ -M_{i,j}, & \text{otherwise} \end{cases} \tag{4}$$

Return $\mathbf{M}$

Algorithm 5 Compression

Input: Matrix $\mathbf{M} \in \mathbb{R}^{\hat{n}\times\hat{m}}$ to be factorized.
Output: Factorized vectors $\mathbf{r} \in \mathbb{R}^{\hat{n}\times 1}$, $\mathbf{c} \in \mathbb{R}^{1\times\hat{m}}$, and binary sign matrix $\mathbf{S} \in \{0,1\}^{\hat{n}\times\hat{m}}$.

$$S_{i,j} = \begin{cases} 1, & \text{if } M_{i,j} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Compute:

$$(\mathbf{r}, \mathbf{c}) = (|\mathbf{M}|\mathbf{1}_{\hat{m}}, \mathbf{1}_{\hat{n}}^{\top}|\mathbf{M}|) \tag{6}$$

Normalize:

$$\mathbf{r} = \begin{cases} \mathbf{r}/(\mathbf{1}_{\hat{n}}^{\top}\mathbf{r}), & \text{if } \hat{n} \leq \hat{m} \\ \mathbf{r}, & \text{otherwise} \end{cases} \tag{7}$$

$$\mathbf{c} = \begin{cases} \mathbf{c}/(\mathbf{c}\mathbf{1}_{\hat{m}}), & \text{if } \hat{n} > \hat{m} \\ \mathbf{c}, & \text{otherwise} \end{cases} \tag{8}$$

Return $\mathbf{r}, \mathbf{c}, \mathbf{S}$

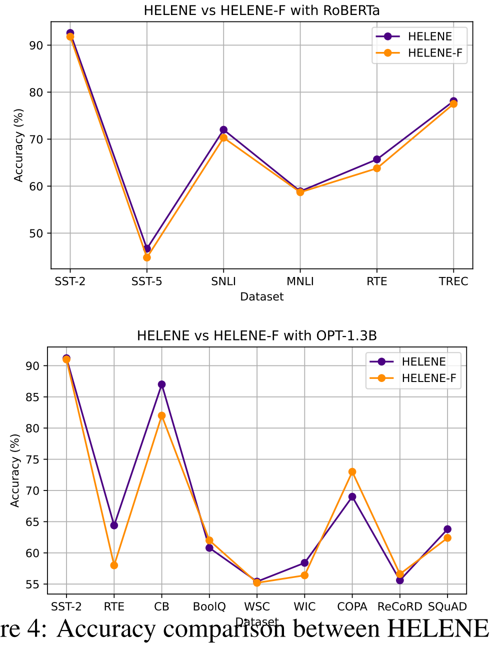


Figure 4: Accuracy comparison between HELENE and HELENE-F on RoBERTa (left) and OPT-1.3B (right).

D Extra Experiments

We assert that HELENE demonstrates robust performance across both small-scale and large-scale models, consistently outperforming MeZO. Specifically, on a 13B model, HELENE achieves an average improvement of **1.5%**. Furthermore, in the domain of non-differentiable optimization, it yields an average performance gain of **4%**.

Task	SST-2	RTE	CB	BoolQ	WSC	WIC	COPA	ReCoRD	SQuAD
Task Type	——— classification ———						— multiple choice —		– generation –
Zero-shot	53.4	53.1	37.5	45.7	44.2	57.0	75.0	70.3	27.1
ICL	80.3	53.1	48.2	58.5	44.2	50.6	69.0	71.0	59.0
LP	80.3	52.7	44.6	58.9	47.1	50.6	69.0	71	75.9
MeZO	89.6	55.8	77.0	59.6	55.0	58.0	74.0	60.0	62.2
MeZO (LoRA)	90.8	63.0	68.6	67.2	51.2	58.0	79.0	59.8	67.6
MeZO (prefix)	92.4	52.8	66.0	61.6	51.6	52.8	74.0	56.8	56.0
HELENE	91.2	**64.4**	**87.0**	60.8	**55.4**	**58.4**	69.0	55.6	63.8
HELENE-F	91.0	58	82.0	62	55.2	56.4	73.0	56.6	62.4
HELENE (LoRA)	91.4	63.5	78.0	**64.0**	53.8	58.0	**82.0**	**60.2**	60.4
HELENE (prefix)	**92.4**	51.6	74.0	62.5	52	57.2	80.0	58.8	**68.4**
FT (12× memory)	90.8	73.4	77	70.2	53	60.2	81.0	59.6	70.9

Table 9: Experiments on OPT-1.3B (with 1000 examples). ICL: in-context learning; LP: linear probing; FT: full-parameter fine-tuning with Adam. We highlight the best results in bold to facilitate comparison.

Task	SST-2	RTE	WSC	WIC	Average
13B MeZO	91.4	66.1	63.5	59.4	70.1
13B HELENE	**92.8**	**68.5**	**64.0**	**61.1**	**71.6**
13B HELENE-F	92.3	67.2	62.8	59.6	70.48

Table 10: Performance comparison across different optimizers on multiple tasks.

Model	RoBERTa-large (350M)				OPT-13B
Task	SST-2	SST-5	SNLI	TREC	SQuAD
Zero-shot	79.0	35.5	50.2	32.0	46.2
MeZO	92.7	48.9	**82.7**	68.6	78.5
HELENE	**93.5**	**51.7**	82.5	**82**	**82.5**

Table 11: Experiments on non-differentiable optimization objectives (F1), using full-parameter tuning for classification ($k = 512$) and prefix tuning for SQuAD (1,000 examples).

E Ablation Study

E.1 Evaluating the Impact of Key Components on Convergence and Stability

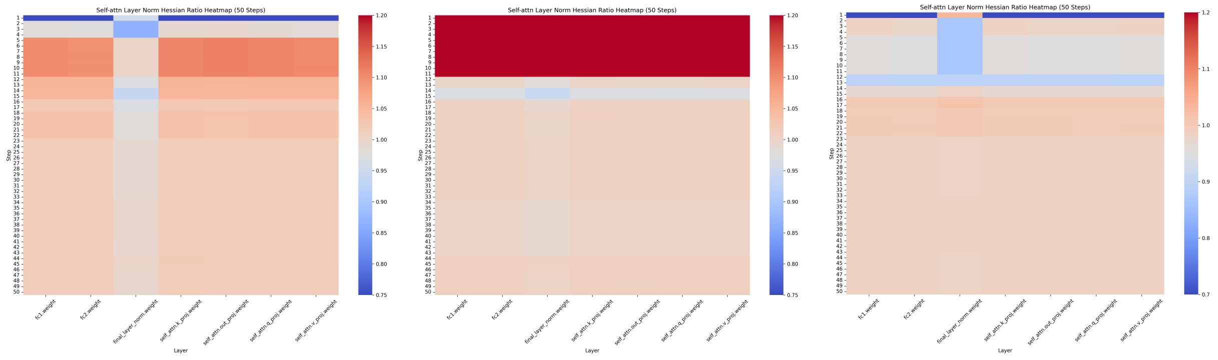
Figure 6 illustrates the effectiveness of each component in our algorithm. Adding momentum to MeZO alone doesn't improve performance. Introducing bias in the gradient boosts convergence speed, but causes loss to increase later in training due to biased gradient estimates. To counter this, we added an annealing term to make the gradient asymptotically unbiased, which stabilizes the loss. Inspired by Sophia, we introduced the clipped Hessian to address heterogeneous curvatures, further improving convergence speed. Our ablation study validates both the motivation and effectiveness of these components.

E.2 Magnitude clipping

From Figure 7, We observed that the Hessian values associated with the variance in the normalization layer are substantially larger than those in other layers (e.g., bias, QKV, and fully-connected layers). Consequently, we recommend using a stricter gradient clipping threshold for the normalization layer—specifically, clipping by norm at 2.0—while adopting a threshold of 1.0 for layers such as QKV. To determine these thresholds, we conducted a grid search for clipping values in [0.8,0.9,1.5,2,2.5,3]. Our results show that setting the threshold to 2.0 for the normalization layer and 1.0 for the QKV layer yields superior performance compared to applying a universal clipping value across all layers.

In Figure 5, the color bar represents the ratio of the standard deviation (STD) of the Hessian values for the `self_attn_layer_norm` layer to the STD of the Hessian values for other layers. The y-axis corresponds to the training steps, while the x-axis represents different layers within a block. For our investigation, we selected the 5th, 10th, and 15th blocks from the `OPT-1.3B` model, which consists of 24 layers.

As shown in the figure 5, at the early stages of training, the STD of the Hessian values for the `self_attn_layer_norm` layer is substantially higher than that of other layers, indicating a greater presence of extreme Hessian values. This suggests that stronger clipping should be applied to the Hessian values of the `self_attn_layer_norm` layer. As training progresses, the ratio between these values stabilizes and gradually approaches 1. This observation indicates the effectiveness of our **layer-wise clipping** strategy. Detailed numerical experiments can be found in the Figure 7.



(a) Block 5 in OPT-1.3B

(b) Block 10 in OPT-1.3B

(c) Block 15 in OPT-1.3B

Figure 5: The standard deviation (STD) of the Hessian values for the `self_attn_layer_norm` layer is substantially higher than that of other layers. As training progresses, the ratio between these values stabilizes and gradually approaches 1. This observation indicates the effectiveness of our layer-wise clipping strategy.

E.3 Investigation into the Convergence Instability of Sophia

We study the reasons for Sophia's failure in the Figure 1 by counting the number of clip triggers. We computed the loss between timesteps 400 and 800, with a mean value of 0.57. The average loss between timesteps 1400 and 1800 was 0.65. We then analyzed the number of times the Sophia clipping mechanism was triggered within these two time intervals. Our analysis covered the Q, K, V matrices, fully connected layers, and bias layers. We found that the frequency of clipping in the interval where the mean loss was 0.65 was 1.18 to 1.22 times higher than in the interval where the mean loss was 0.57.

Based on these experimental observations, we conclude that Sophia's clipping mechanism tends to be over-triggered in complex data scenarios, particularly when faced with heterogeneous curvature. This over-triggering can result in non-convergence, aligning with our intuition. In the zeroth-order setting, gradients are estimated using SPSA, and excessive clipping of the $\frac{g}{H}$ terms can lead to instability and failure of the model to converge.

F Detailed Convergence Analysis

Lemma 1 (Divergence to Infinity)**.** *Under Assumption 1, for each layer i in a neural network model, assume the function $L : \mathbb{R}^{d_i} \to \mathbb{R}$ is strictly convex, twice continuously differentiable, and has a unique minimizer denoted by $\boldsymbol{\theta}_i^*$. For any parameter vector $\boldsymbol{\theta}_i$ of layer i such that $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \geq 1$, the function $L(\boldsymbol{\theta}_i)$ diverges to infinity as $\|\boldsymbol{\theta}_i\|_2 \to \infty$.*

Proof. By the strict convexity of L, for any $\boldsymbol{\theta}_i$ such that $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \geq 1$, we have:

$$\frac{L(\boldsymbol{\theta}_i) - L(\boldsymbol{\theta}_i^*)}{\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2} \geq \min_{\|\phi\|_2=1} L(\boldsymbol{\theta}_i^* + \phi) - L(\boldsymbol{\theta}_i^*), \quad (9)$$

where ϕ is a unit vector. For the convenience, here $L(\boldsymbol{\theta}_i)$ denotes $L(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{-i})$ where $\boldsymbol{\theta}_{-i}$ denotes the parameters in the whole model except $\boldsymbol{\theta}_i$, and $L(\boldsymbol{\theta}_i^*)$ denotes $L(\boldsymbol{\theta}_i^*|\boldsymbol{\theta}_{-i}^*)$. Define Δ_i as:

$$\Delta_i = \min_{\|\phi\|_2=1} L(\boldsymbol{\theta}_i^* + \phi) - L(\boldsymbol{\theta}_i^*), \quad (10)$$

a positive constant due to the strict convexity of L indicating the minimal rate of increase of L around $\boldsymbol{\theta}_i^*$.

Thus, the inequality can be written as:

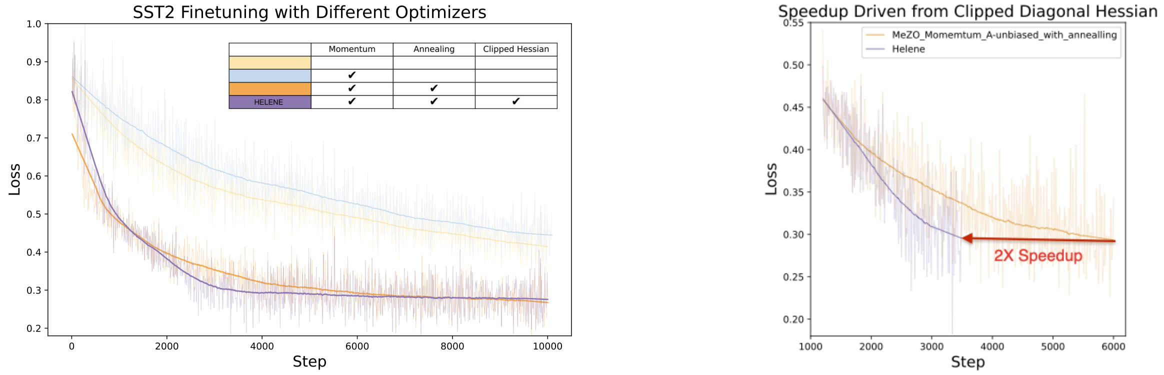
$$L(\boldsymbol{\theta}_i) \geq \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \Delta_i + L(\boldsymbol{\theta}_i^*). \quad (11)$$

This implies that as $\|\boldsymbol{\theta}_i\|_2 \to \infty$, which necessarily implies $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \to \infty$, the loss $L(\boldsymbol{\theta}_i)$ also diverges to infinity, since the term $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \Delta_i$ dominates and increases without bound. □

Note: We do not require the Hessian of the loss function, $\nabla^2 L(\boldsymbol{\theta}_i)$, to be Lipschitz continuous; Assumption 2 only requires that the Hessian is continuous in a multiplicative sense within a neighborhood of constant radius.

Lemma 2 (Parameter Bound)**.** *Let $L : \mathbb{R}^d \to \mathbb{R}$ be a loss function for a neural network composed of multiple layers, each with parameters $\boldsymbol{\theta}_i$, and L is twice continuously differentiable and strictly convex with respect to each layer's parameters at a global minimizer $\boldsymbol{\theta}^*$. Assume each layer i satisfies the following condition:*

$$L(\boldsymbol{\theta}_i) - \min L \leq \frac{\mu_i R_i^2}{4},$$



(a) Ablation study for key components of HELENE. The effectiveness of annealing and the clipped diagonal Hessian are demonstrated progressively from high to low.

(b) The image on the right is a zoomed-in view of the left one, showing that our proposed HELENE is twice as fast as the compared methods.

Figure 6: Comparison of tuning processes and ablation studies with different optimization algorithms.

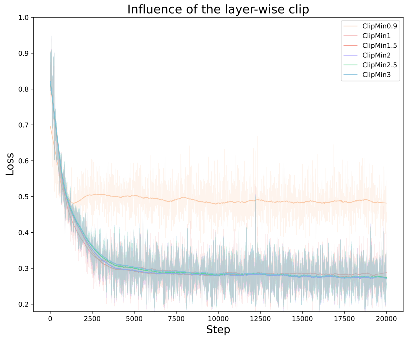


Figure 7: The performance and convergence of HELENE when fine-tuning the OPT-1.3B model on SST-2 using various clipping lower bounds. In these experiments, 'ClipMin' denotes the clipping value for the QKV layers, while the clipping value for the normalization layer is fixed at 2.

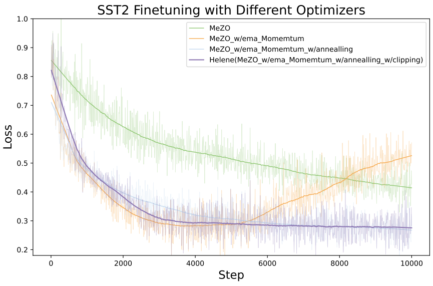


Figure 8: Gradient Descent methods, SPSA gradient estimation becomes increasingly noisy as optimization progresses.

where μ_i is the minimum eigenvalue of the Hessian of L at the minimizer for parameters of layer i, and R_i is a predefined radius. Then, it holds that

$$\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \le R_i.$$

Proof. Suppose, by way of contradiction, there exists a $\boldsymbol{\theta}_i$ such that $L(\boldsymbol{\theta}_i) \le \frac{\mu_i R_i^2}{4}$, but $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 > R_i$. Define $\boldsymbol{\theta}_i'$ as:

$$\boldsymbol{\theta}_i' = \boldsymbol{\theta}_i^* + \sqrt{2(L(\boldsymbol{\theta}_i) - \min L)} \frac{\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*}{\mu_i \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2}. \tag{12}$$

Since $\boldsymbol{\theta}_i'$ is a point between $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_i^*$, and due to the strict convexity of L, we have $L(\boldsymbol{\theta}_i') < L(\boldsymbol{\theta}_i)$ by convexity. Considering the Taylor expansion of L at $\boldsymbol{\theta}_i^*$ along the direction towards $\boldsymbol{\theta}_i'$, we have:

$$f(t) = L(\boldsymbol{\theta}_i^* + t(\boldsymbol{\theta}_i' - \boldsymbol{\theta}_i^*)),$$

$$f(1) = L(\boldsymbol{\theta}_i'), \quad f(0) = L(\boldsymbol{\theta}_i^*), \quad f'(0) = 0,$$

$$f''(t) = (\boldsymbol{\theta}_i' - \boldsymbol{\theta}_i^*)^T \nabla^2 L(t\boldsymbol{\theta}_i' + (1-t)\boldsymbol{\theta}_i^*)(\boldsymbol{\theta}_i' - \boldsymbol{\theta}_i^*),$$

Given $f''(t) \ge \frac{\mu_i}{2}\|\boldsymbol{\theta}_i' - \boldsymbol{\theta}_i^*\|_2^2$ from Assumption 2 and the convexity, the Taylor expansion yields:

$$\begin{aligned} f(1) \ge & f(0) + f'(0) + \frac{1}{2} f''(t) \\ = & L(\boldsymbol{\theta}_i^*) + \frac{\mu_i}{2}\|\boldsymbol{\theta}_i' - \boldsymbol{\theta}_i^*\|_2^2, \end{aligned}$$

thus,

$$L(\boldsymbol{\theta}_i') \ge L(\boldsymbol{\theta}_i^*) + \frac{\mu_i}{2}\|\boldsymbol{\theta}_i' - \boldsymbol{\theta}_i^*\|_2^2$$

which contradicts the assumption that $L(\boldsymbol{\theta}_i') < L(\boldsymbol{\theta}_i)$. Therefore, the original assumption that $|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 > R_i$ must be false, concluding that$\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \le R_i$. □

Lemma 3 (Gradient Norm Bound)**.** *For any $\boldsymbol{\theta}_i$ in layer i of a neural network, satisfying $\|\nabla L(\boldsymbol{\theta}_i)\|_2 \le \frac{\mu_i R_i}{2}$, where μ_i is the minimum eigenvalue of the Hessian of L at the minimizer for layer i parameters, it holds that $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \le R_i$.*

Proof. Assume, by way of contradiction, there exists a $\boldsymbol{\theta}_i$ with $\|\nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{\mu_i R_i}{2}$ and $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 > R_i$. Define a function $f(t)$ by:

$$f(t) = \nabla L(\boldsymbol{\theta}_i^* + t\cdot(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*)) \cdot \frac{\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*}{\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2}, \quad (13)$$

where $f(0) = \nabla L(\boldsymbol{\theta}_i^*)$ due to $\boldsymbol{\theta}_i^*$ being a minimizer, and $f(R_i) = \nabla L(\boldsymbol{\theta}_i)$.

Due to the strict convexity of L, $f(t)$ is a strictly monotone increasing function. The derivative with respect to t, must satisfy:

$$\begin{aligned} f'(t) = \frac{d}{dt}\Bigg(\nabla L\Big(\boldsymbol{\theta}_i^* + t\cdot(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*)\Big)\cdot \\ \frac{\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*}{\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2}\Bigg) \quad (14) \\ \geq \frac{\|\nabla L(\boldsymbol{\theta}_i)\|_2}{2}. \quad (15) \end{aligned}$$

by Assumption 2 and the fact that the gradient norm does not increase more than twice in any direction within the ball of radius R_i.

The fundamental theorem of calculus and the above inequality imply:

$$\begin{aligned} f(R_i) &= f(0) + \int_0^{R_i} f'(t)dt \\ &\geq \int_0^{R_i} \frac{\|\nabla L(\boldsymbol{\theta}_i)\|_2}{2}dt = \frac{R_i\|\nabla L(\boldsymbol{\theta}_i)\|_2}{2}, \end{aligned}$$

However, $f(R_i) = \|\nabla L(\boldsymbol{\theta}_i)\|_2$ and this leads to

$$\|\nabla L(\boldsymbol{\theta}_i)\|_2 \geq \frac{R_i\|\nabla L(\boldsymbol{\theta}_i)\|_2}{2}, \quad (16)$$

a contradiction unless $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \leq R_i$.

Therefore, the original assumption that $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 > R_i$ must be false, proving the lemma. □

Lemma 4 (Stability of Gradient Flow)**.** *Suppose the gradient $\nabla L(\boldsymbol{\theta}_i(t))$ and the Hessian $\nabla^2 L(\boldsymbol{\theta}_i(t))$ of the loss function L satisfy the conditions for all $t \in [0,1]$ that ensure stability and convergence to a minimizer $\boldsymbol{\theta}_i^*$. Assume the differential equation*

$$\begin{aligned} &\frac{d\boldsymbol{\theta}_i(t)}{dt} = -\left(\nabla^2 L(\boldsymbol{\theta}_i(t))\right)^{-1}\nabla L(\boldsymbol{\theta}_i(t)), \\ &\boldsymbol{\theta}_i(0) = \boldsymbol{\theta}_i, \quad \boldsymbol{\theta}_i(1) = \boldsymbol{\theta}_i^*, \end{aligned}$$

has at least one solution on the interval [0, 1] and satisfies $\nabla L(\boldsymbol{\theta}_i(t)) = (1-t)\nabla L(\boldsymbol{\theta}_i)$ for all $t \in [0,1]$.

Proof. We demonstrate this by showing that the given ordinary differential equation (ODE) is well-posed under the assumptions. The initial value problem

$$\frac{d\boldsymbol{\theta}_i(t)}{dt} = -\left(\nabla^2 L(\boldsymbol{\theta}_i(t))\right)^{-1}\nabla L(\boldsymbol{\theta}_i(t)), \quad (17)$$

can be solved over the interval [0, 1] due to the continuity and positive definiteness of $\nabla^2 L$, which ensures the existence and uniqueness of the solution by the Picard-Lindelöf theorem.

Define $T_{\max,i}$ as the largest positive number such that the solution exists on $[0, T_{\max,i}]$. We claim $T_{\max,i} \geq 1$, based on the behavior of the gradient along the solution path. Considering:

$$\begin{aligned} \frac{d}{dt}\nabla L(\boldsymbol{\theta}_i(t)) &= \nabla^2 L(\boldsymbol{\theta}_i(t))\frac{d\boldsymbol{\theta}_i(t)}{dt} \\ &= -\nabla L(\boldsymbol{\theta}_i(t)). \end{aligned} \quad (18)$$

which implies that $\nabla L(\boldsymbol{\theta}_i(t)) = e^{-t}\nabla L(\boldsymbol{\theta}_i)$. Since $\nabla L(\boldsymbol{\theta}_i(t)) = (1-t)\nabla L(\boldsymbol{\theta}_i)$ for $t \in [0,1]$, the condition aligns perfectly.

Finally, since $\boldsymbol{\theta}_i(1)$ has zero gradient by the construction of the ODE, $\boldsymbol{\theta}_i(1)$ must be $\boldsymbol{\theta}_i^*$. This completes the proof. □

Lemma 5 (Quadratic Form Integration)**.** *Assume the gradient norm $\|\nabla L(\boldsymbol{\theta}_i)\|_2$ and the Hessian $\nabla^2 L(\boldsymbol{\theta}_i)$ satisfy certain conditions over the interval $[0,1]$. Suppose either*

1. $L(\boldsymbol{\theta}_i) - \min L \leq \frac{\mu_i R_i^2}{16}$, *or*
2. $\|\nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{\mu_i R_i}{4}$,

where μ_i is the minimum eigenvalue of the Hessian at the minimizer for the parameters of layer i, then it holds that

$$\begin{aligned} &\left\|\nabla L(\boldsymbol{\theta}_i)^T(\nabla^2 L(\boldsymbol{\theta}_i))^{-1}\nabla L(\boldsymbol{\theta}_i)\right\| \\ \leq &4(L(\boldsymbol{\theta}_i) - \min L). \end{aligned}$$

Proof. Let $\{\boldsymbol{\theta}_i(t)\}_{t=0}^1$ be the solution of the following differential equation:

$$\begin{aligned} &\frac{d\boldsymbol{\theta}_i(t)}{dt} = -(\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1}\nabla L(\boldsymbol{\theta}_i(t)), \\ &\boldsymbol{\theta}_i(0) = \boldsymbol{\theta}_i, \quad \boldsymbol{\theta}_i(1) = \boldsymbol{\theta}_i^*. \end{aligned}$$

From Lemma 4, adapted for each layer, we have $\nabla L(\boldsymbol{\theta}_i(t)) = (1-t)\nabla L(\boldsymbol{\theta}_i)$ for all $t \in [0,1]$. Assume $\|\boldsymbol{\theta}_i(t) - \boldsymbol{\theta}_i^*\| \leq R_i/2$ by Lemmas 2 and 3.

By Assumption 2, for each layer, this implies:

$$(\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1} \geq \frac{1}{2}(\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \tag{19}$$

for all $t \in [0, 1]$.

Integrating the quadratic form along the path, let $g_i := \nabla L(\boldsymbol{\theta}_i)$, $H_i(t) := \nabla^2 L(\boldsymbol{\theta}_i(t))$, we have:

$$\begin{aligned} L(\boldsymbol{\theta}_i) - \min L &= L\big(\boldsymbol{\theta}_i(0)\big) - L\big(\boldsymbol{\theta}_i(1)\big) \\ &= \int_0^1 (1-t)^2\, g_i^\top\, H_i(t)^{-1}\, g_i \,\mathrm{d}t. \end{aligned} \tag{20}$$

Substituting the inequality from equation 19, we simplify:

$$\begin{aligned} &\frac{1}{2}\int_0^1 (1-t)^2 dt (\nabla L(\boldsymbol{\theta}_i))^T (\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \nabla L(\boldsymbol{\theta}_i) \\ =&\frac{1}{6}(\nabla L(\boldsymbol{\theta}_i))^T (\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \nabla L(\boldsymbol{\theta}_i). \end{aligned}$$

This integration shows that $\|\nabla L(\boldsymbol{\theta}_i)^T (\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \nabla L(\boldsymbol{\theta}_i)\| \leq 4(L(\boldsymbol{\theta}_i) - \min L)$, completing the proof. □

Lemma 6 (Gradient and Loss Bound)**.** *Assuming the gradient norm $\|\nabla L(\boldsymbol{\theta}_i)\|_2$ and the conditions on the loss function L are such that either*

1. $L(\boldsymbol{\theta}_i) - \min L \leq \frac{\mu_i R_i^2}{4}$, *or*
2. $\|\nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{R_i \mu_i}{2}$,

it holds that

$$L(\boldsymbol{\theta}_i) - \min L \leq \frac{1}{\mu_i}\|\nabla L(\boldsymbol{\theta}_i)\|^2. \tag{21}$$

Proof. The proof follows a reasoning similar to that of Lemma 5 but adapted for each layer. Given the conditions on $L(\boldsymbol{\theta}_i) - \min L$ or the norm of the gradient $\|\nabla L(\boldsymbol{\theta}_i)\|_2$, we utilize the connection between the gradient norm and the difference in loss to bound $L(\boldsymbol{\theta}_i) - \min L$.

From the gradient norm bound $\|\nabla L(\boldsymbol{\theta}_i)\|_2$ and the positive definiteness and continuity of $\nabla^2 L$, the loss function exhibits quadratic behavior near the minimizer. This is characterized by the Taylor expansion:

$$L(\boldsymbol{\theta}_i) \approx L(\boldsymbol{\theta}_i^*) + \frac{1}{2}(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*)^T \nabla^2 L(\boldsymbol{\theta}_i^*)(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*),$$

where $\boldsymbol{\theta}_i^*$ is the minimizer of L.

Using the bound $\|\nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{R_i \mu_i}{2}$, the Taylor series expansion around $\boldsymbol{\theta}_i^*$ implies:

$$L(\boldsymbol{\theta}_i) - \min L \leq \frac{1}{2\mu_i}\|\nabla L(\boldsymbol{\theta}_i)\|^2, \tag{22}$$

satisfying the condition given by Lemma 6.

This completes the proof by relating the behavior of the loss function's gradient at $\boldsymbol{\theta}_i$ to its minimum value, leveraging the quadratic approximation provided by the Hessian at the minimizer. □

Lemma 7 (Norm Bound on Inverse Hessian and Gradient Product)**.** *Assuming the gradient $\nabla L(\boldsymbol{\theta}_i)$ and the Hessian $\nabla^2 L(\boldsymbol{\theta}_i)$ satisfy certain conditions such that either*

1. $L(\boldsymbol{\theta}_i) - \min L \leq \frac{\mu_i R_i^2}{16}$, *or*
2. $\|\nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{R_i \mu_i}{4}$,

it holds that

$$\|\nabla^2 L(\boldsymbol{\theta}_i)^{-1} \nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{8(L(\boldsymbol{\theta}_i) - \min L)}{\mu_i}.$$

Proof. We derive this by using the properties of the Hessian and the gradient for the loss function L specific to layer i. From Lemma 2, we have:

$$\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^*\|_2 \leq R_i.$$

Given that $\nabla^2 L(\boldsymbol{\theta}_i^*) \geq \frac{\mu_i}{2} I$, and from Lemma 5 adapted for layer i, it holds that:

$$\begin{aligned} &4(L(\boldsymbol{\theta}_i) - \min L) \geq \\ &\qquad \|\nabla L(\boldsymbol{\theta}_i)^T (\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \nabla L(\boldsymbol{\theta}_i)\| \end{aligned} \tag{23}$$

Expanding and manipulating the inequality, we derive:

$$\begin{aligned} &\|\nabla L(\boldsymbol{\theta}_i)\|^T (\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \|\nabla L(\boldsymbol{\theta}_i)\| \\ =&\|\nabla^2 L(\boldsymbol{\theta}_i)^{-1} \nabla L(\boldsymbol{\theta}_i)\|^2. \end{aligned}$$

Given $\nabla^2 L(\boldsymbol{\theta}_i)^{-1} \leq \frac{2}{\mu_i} I$, we can substitute this into our calculation to find:

$$\begin{aligned} \|\nabla^2 L(\boldsymbol{\theta}_i)^{-1} \nabla L(\boldsymbol{\theta}_i)\|^2 &\leq \frac{2}{\mu_i}\|\nabla L(\boldsymbol{\theta}_i)\|^2 \\ &\leq \frac{4(L(\boldsymbol{\theta}_i) - \min L)}{\mu_i}, \end{aligned}$$

and finally,

$$\|\nabla^2 L(\boldsymbol{\theta}_i)^{-1} \nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{8(L(\boldsymbol{\theta}_i) - \min L)}{\mu_i},$$

completing the proof. □

Lemma 8. *For any $\boldsymbol{\theta}_i \in \mathbb{R}^{d_i}$, where $\boldsymbol{\theta}_i$ represents the parameters for the i-th layer, and satisfying that*

$$\|\nabla^2 L(\boldsymbol{\theta}_i)^{-1} \nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{R}{2},$$

it holds that

$$\begin{aligned} L(\boldsymbol{\theta}_i) - \min L \leq& \nabla L(\boldsymbol{\theta}_i)^T (\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \nabla L(\boldsymbol{\theta}_i) \\ \leq& 4(L(\boldsymbol{\theta}_i) - \min L). \end{aligned}$$

Proof. Let $\{\boldsymbol{\theta}_i(t)\}_{t=0}^1$ be the solution of the following differential equation:

$$\frac{d\boldsymbol{\theta}_i(t)}{dt} = -(\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1}\nabla L(\boldsymbol{\theta}_i(t)),$$

where $\boldsymbol{\theta}_i(0) = \boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_i(1) = \boldsymbol{\theta}_i^*$.

We claim that for all $t \in [0, 1]$, $\|\boldsymbol{\theta}_i(t) - \boldsymbol{\theta}_i\|_2 \leq R_i$. If not, let T be the smallest positive number such that $\|\boldsymbol{\theta}_i(T) - \boldsymbol{\theta}_i\|_2 = R_i$. Such T exists because $\|\boldsymbol{\theta}_i(t) - \boldsymbol{\theta}_i\|_2$ is continuous in t and $\|\boldsymbol{\theta}_i(0) - \boldsymbol{\theta}_i\|_2 = 0$.

We can now bound the distance:

$$R_i = \|\boldsymbol{\theta}_i(T) - \boldsymbol{\theta}_i(0)\|_2 \leq \int_0^T \left\|\frac{d\boldsymbol{\theta}_i(t)}{dt}\right\|_2 dt.$$

Substituting the derivative expression for $\boldsymbol{\theta}_i(t)$, we get:

$$= \int_0^T \left\|(\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1}\nabla L(\boldsymbol{\theta}_i(t))\right\|_2 dt$$

$$\leq \int_0^T \|\nabla^2 L(\boldsymbol{\theta}_i(t))^{-1}\|_2 \|\nabla L(\boldsymbol{\theta}_i(t))\|_2 dt.$$

From Assumption 2, we know that:

$$\nabla^2 L(\boldsymbol{\theta}_i)^{-1} \leq 2(\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1}.$$

Thus, we can bound this integral:

$$\leq 2\int_0^T \|\nabla^2 L(\boldsymbol{\theta}_i(t))^{-1}\nabla L(\boldsymbol{\theta}_i(t))\|_2 dt$$

$$\leq 2T\|\nabla^2 L(\boldsymbol{\theta}_i(t))^{-1}\nabla L(\boldsymbol{\theta}_i(t))\|_2.$$

Using the assumption that $\|\nabla^2 L(\boldsymbol{\theta}_i)^{-1}\nabla L(\boldsymbol{\theta}_i)\|_2 \leq \frac{R_i}{2}$, we get:

$$\leq 2T\frac{R_i}{2} = R_i T,$$

which implies that $T = 1$.

Therefore, for all $t \in [0, 1]$, $\|\boldsymbol{\theta}_i(t) - \boldsymbol{\theta}_i\|_2 \leq R_i$. By Assumption 2, we also have:

$$2(\nabla^2 L(\boldsymbol{\theta}_i))^{-1} \preceq (\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1} \preceq \frac{1}{2}(\nabla^2 L(\boldsymbol{\theta}_i))^{-1}.$$

Now, we compute the difference in the loss function:

$$\begin{aligned}
&L(\boldsymbol{\theta}_i) - \min L \\
=&L(\boldsymbol{\theta}_i(0)) - L(\boldsymbol{\theta}_i(1)) \\
=&\int_0^1 \nabla L(\boldsymbol{\theta}_i(t))^T(\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1}\nabla L(\boldsymbol{\theta}_i(t))dt \\
=&\int_0^1 (1-t)\nabla L(\boldsymbol{\theta}_i)^T(\nabla^2 L(\boldsymbol{\theta}_i))^{-1}\nabla L(\boldsymbol{\theta}_i)dt.
\end{aligned}$$

Thus:

$$L(\boldsymbol{\theta}_i) - \min L \leq \frac{1}{2}\nabla L(\boldsymbol{\theta}_i)^T(\nabla^2 L(\boldsymbol{\theta}_i))^{-1}\nabla L(\boldsymbol{\theta}_i).$$

Finally, using the fact that $\int_0^1 (1-t)dt = \frac{1}{2}$, we complete the proof, showing that:

$$\begin{aligned}
&L(\boldsymbol{\theta}_i) - \min L \\
\leq&\nabla L(\boldsymbol{\theta}_i)^T(\nabla^2 L(\boldsymbol{\theta}_i))^{-1}\nabla L(\boldsymbol{\theta}_i) \\
\leq&4(L(\boldsymbol{\theta}_i) - \min L).
\end{aligned}$$

□

Lemma 9. *If $\lambda_i \leq \frac{R_i}{2\sqrt{d_i}}$, then for any $\Delta \leq \frac{R_i\mu}{10}$ and any $\boldsymbol{\theta}_i \in \mathbb{R}_{\beth}{}^{d_i}$ satisfying*

$$\sum_{i=1}^{d_i} \min\left\{\boldsymbol{p}_i^T\nabla L(\boldsymbol{\theta}_i)\sigma_i^{-1}\boldsymbol{p}_i^T\nabla L(\boldsymbol{\theta}_i)\right\} \leq \Delta,$$

where $\nabla^2 L(\boldsymbol{\theta}_i) = V_i\Sigma_i V_i^T$ is the eigen-decomposition of $\nabla^2 L(\boldsymbol{\theta}_i)$, $\boldsymbol{p}_i$ is the i-th row of V_i, and $\Sigma_i = diag(\sigma_1, \ldots, \sigma_{d_i})$, it holds that

$$L(\boldsymbol{\theta}_i) - \min L \leq \frac{25\Delta^2}{\lambda_i^2\mu}.$$

Proof. Let $\{\boldsymbol{\theta}_i(t)\}_{t=0}^1$ be the solution to the ODE

$$\frac{d\boldsymbol{\theta}_i(t)}{dt} = -(\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1}\nabla L(\boldsymbol{\theta}_i(t)),$$

starting from $\boldsymbol{\theta}_i(0) = \boldsymbol{\theta}_i$ and assume $\boldsymbol{\theta}_i(1) = \boldsymbol{\theta}_i^*$ as derived in previous lemmas.

By Lemma 2, $\|\boldsymbol{\theta}_i(t) - \boldsymbol{\theta}_i^*\|_2 \leq R_i$ for all $t \in [0, 1]$. Define $I_0 \subseteq [d_i]$ as the indices where clipping does not occur. We have:

$$\sum_{i\in I_0} \sigma_i^{-1}\left|\boldsymbol{p}_i^T\nabla L(\boldsymbol{\theta}_i)\right|^2 \leq \Delta.$$

Using Assumption 2, the Hessian continuity within a local radius implies:

$$\sum_{i\in I_0} \left|\boldsymbol{p}_i^T\nabla L(\boldsymbol{\theta}_i(t))\right|^2 \leq \Delta.$$

For the newly restricted convex function L_0 on $\mathbb{R}_i^{I_0}$, which is L restricted to the subspace of $\mathbb{R}_i^{d_i}$ spanned by vectors corresponding to I_0, by Lemma 1 and assuming L_0 is strictly convex, we apply Lemmas 6 and 8 by restricting to I_0:

$$\begin{aligned}
&\|\nabla L_0(\boldsymbol{\theta}_i) + V_{I_0}^T\boldsymbol{\theta}_i^*\|_2^2 \\
=&\|\nabla L_0(\boldsymbol{\theta}_i)\|_2^2 \leq \mu^{-1}\|\nabla L_0(\boldsymbol{\theta}_i)\|_2^2 \leq \frac{25\Delta^2}{\lambda_i^2\mu}.
\end{aligned}$$

Integrating the differential for L_0, we can show:

$$
\begin{aligned}
&L(\boldsymbol{\theta}_i) - \min L \\
\leq &\int_0^1 \nabla L(\boldsymbol{\theta}_i(t))^T (\nabla^2 L(\boldsymbol{\theta}_i(t)))^{-1} \nabla L(\boldsymbol{\theta}_i(t)) dt \\
\leq &\frac{25\Delta^2}{\lambda_i^2 \mu}.
\end{aligned}
$$

This completes the proof. □

Lemma 10 (Descent Lemma). *For any $\eta > 0$ and per-layer $\lambda_i > 0$ with $\eta\lambda_i \leq \frac{R_i}{\sqrt{d_i}}$, define*

$$\boldsymbol{\theta}_i^+ = \boldsymbol{\theta}_i - \eta clip\left((\hat{\boldsymbol{g}}_i \hat{\boldsymbol{g}}_i)^{-1} \hat{\boldsymbol{g}}_i, \lambda_i\right),$$

where $\hat{\boldsymbol{g}}_i$ is the estimated gradient using a ZO finite difference method with noise ϵ, such that:

$$\hat{\boldsymbol{g}}_i = \nabla L(\boldsymbol{\theta}_i) + \epsilon.$$

The theoretical bound for the descent is given by:

$$
\begin{aligned}
&L(\boldsymbol{\theta}_i^+) - L(\boldsymbol{\theta}_i) \\
\leq &-(\eta - \eta^2 \beta_i \lambda_i) \sum_{j=1}^{d_i} \min\{\lambda_i, \frac{1}{\sigma_{i,j}} |\mathbf{v}_{i,j}^T \nabla L(\boldsymbol{\theta}_i)|^2 \\
&+ C(\delta_g^2 + \delta_H^2)\} \\
\leq &-(\eta - \eta^2) \sum_{i=1}^{d} \min\left\{\lambda_i \, |\hat{\boldsymbol{g}}_i|, \, (\hat{\boldsymbol{g}}_i \hat{\boldsymbol{g}}_i)^{-1} |\hat{\boldsymbol{g}}_i|^2\right\} \\
&+ O(h) + O(1/\sqrt{m}),
\end{aligned}
$$

where h is the step size of the finite difference and m is the number of perturbations performed for finite difference estimation. The second inequality goes equal when h and $1/sqrt(m)$ are sufficiently small.

Proof. Step 1. **Derivation of the upper bound for $\|\hat{\boldsymbol{g}}_i - \nabla L(\boldsymbol{\theta}_i)\|$.** To derive a theoretical bound for $\|\hat{\boldsymbol{g}}_i - \nabla L(\boldsymbol{\theta}_i)\|$, where $\hat{\boldsymbol{g}}_i$ is the gradient estimated using our proposed zero-order method, and $\nabla L(\boldsymbol{\theta}_i)$ is the true gradient, we need to quantify the error due to using finite perturbations to approximate the gradient. Let's denote this error by ϵ, such that:

$$\epsilon_i = \hat{\boldsymbol{g}}_i - \nabla L(\boldsymbol{\theta}_i)$$

Specifically, the gradient estimate for dimension i is obtained by:

$$\hat{\boldsymbol{g}}_i = \frac{1}{m} \sum_{k=1}^{m} \frac{L(\boldsymbol{\theta} + h\boldsymbol{u}_k) - L(\boldsymbol{\theta})}{h} \boldsymbol{u}_k^{(i)},$$

where m is the number of perturbations,h is the step size for finite differences, and $\boldsymbol{u}_k^{(i)}$ represents the i-th component of the random vector $\boldsymbol{u}_k$. The true gradient, on the other hand, is:

$$\nabla L(\boldsymbol{\theta}_i) = \lim_{h \to 0} \frac{L(\boldsymbol{\theta} + h\boldsymbol{u}_k) - L(\boldsymbol{\theta})}{h} \boldsymbol{u}_k^{(i)},$$

The error between the estimated gradient $\hat{\boldsymbol{g}}_i$ and the true gradient $\nabla L(\boldsymbol{\theta}_i)$ arises from two main sources. To derive a theoretical bound for the estimation error, $\|\hat{\boldsymbol{g}}_i - \nabla L(\boldsymbol{\theta}_i)\|$, we consider both sources of error.

1. Finite Difference Approximation Error. By Taylor expansion, for a small step size h, we have:

$$
\begin{aligned}
&L(\boldsymbol{\theta} + h\boldsymbol{u}_k) \\
=&L(\boldsymbol{\theta}) + h\nabla L(\boldsymbol{\theta})^T \boldsymbol{u}_k + \frac{h^2}{2} \boldsymbol{u}_k^T H(\boldsymbol{\theta}) \boldsymbol{u}_k + O(h^3),
\end{aligned}
$$

where $H(\boldsymbol{\theta})$ is the Hessian of L at $\boldsymbol{\theta}$. Thus, the error due to finite differences is of order $O(h)$. Specifically, the bias in the gradient estimate is proportional to:

$$\text{Bias} = O\left(\frac{h}{2} \|H(\boldsymbol{\theta})\|\right).$$

2. Monte Carlo Sampling Error. The gradient estimate involves averaging over m samples of random perturbations. By the Central Limit Theorem, the variance of the gradient estimate decreases with the number of samples m. Specifically:

$$\text{Variance} = O\left(\frac{\sigma^2}{m}\right),$$

where σ^2 is the variance of the directional derivative $\nabla L(\boldsymbol{\theta})^T \boldsymbol{u}_k$.

The total error can be expressed as a combination of the bias and variance components. Using a norm (e.g., Euclidean norm) to quantify the error, we have:

$$\|\hat{\boldsymbol{g}}_i - \nabla L(\boldsymbol{\theta}_i)\| \leq O\left(h\|H(\boldsymbol{\theta})\|\right) + O\left(\frac{\sigma}{\sqrt{m}}\right).$$

Thus, the theoretical bound on the error is:

$$\|\hat{\boldsymbol{g}}_i - \nabla L(\boldsymbol{\theta}_i)\| = O\left(h\|H(\boldsymbol{\theta})\| + \frac{\sigma}{\sqrt{m}}\right).$$

Step 2. **Derivation of the upper bound for $\|\hat{\boldsymbol{g}}_i^2 - \mathbf{diag}(\nabla^2 L(\boldsymbol{\theta}_i))\|$.** Let's denote

$\mathrm{diag}(\nabla^2 L(\boldsymbol{\theta}_i))$ as the diagonal of the true Hessian, and $\hat{H}_i = \hat{\boldsymbol{g}}_i^2$ as the diagonal Hessian estimated from the zero-order gradient estimate, where each diagonal element is given by $\hat{\boldsymbol{g}}_i \hat{\boldsymbol{g}}_i$.

To derive the theoretical bound for $\|\hat{H}_i - H_i\|$, we consider:

$$\|\hat{H}_i - H_i\| = \|\hat{\boldsymbol{g}}_i^2 - \mathrm{diag}(\nabla^2 L(\boldsymbol{\theta}_i))\|.$$

Let's rewrite $\hat{\boldsymbol{g}}_i$ as:

$$\hat{\boldsymbol{g}}_i = \nabla L(\boldsymbol{\theta}_i) + \epsilon_i,$$

where ϵ_i represents the noise introduced due to the limited number of perturbations.

The estimated diagonal Hessian element for each component i can be written as:

$$\hat{H}_i^{(i)} = (\nabla L(\boldsymbol{\theta}_i) + \epsilon_i)^2.$$

Expanding this expression gives:

$$\hat{H}_i^{(i)} = (\nabla L(\boldsymbol{\theta}_i))^2 + 2\nabla L(\boldsymbol{\theta}_i)\epsilon_i + \epsilon_i^2.$$

The true diagonal Hessian element is:

$$H_i^{(i)} = \mathrm{diag}(\nabla^2 L(\boldsymbol{\theta}_i))^{(i)}.$$

Thus, the error for each component can be expressed as:

$$\hat{H}_i^{(i)} - H_i^{(i)} = (\nabla L(\boldsymbol{\theta}_i))^2 + 2\nabla L(\boldsymbol{\theta}_i)\epsilon_i + \epsilon_i^2 - H_i^{(i)}.$$

To find the bound for the error, we need to bound the terms involving ϵ_i:

1. Term 1: $2\nabla L(\boldsymbol{\theta}_i)\epsilon_i$

This term represents the interaction between the true gradient and the noise. Since $\|\epsilon_i\| \leq O\left(h\|H(\boldsymbol{\theta}_i)\| + \frac{\sigma}{\sqrt{m}}\right)$, we can bound this term as:

$$|2\nabla L(\boldsymbol{\theta}_i)\epsilon_i| \leq 2\|\nabla L(\boldsymbol{\theta}_i)\| O\left(h\|H(\boldsymbol{\theta}_i)\| + \frac{\sigma}{\sqrt{m}}\right).$$

2. Term 2: ϵ_i^2

The noise squared term can be bounded by:

$$\epsilon_i^2 \leq O\left(h^2\|H(\boldsymbol{\theta}_i)\|^2 + \frac{\sigma^2}{m}\right).$$

Combining these results, we have:

$$\begin{aligned}&\|\hat{H}_i - H_i\| \\ =&O\left((\nabla L(\boldsymbol{\theta}_i))^2 - H_i + \left(h^2\|H(\boldsymbol{\theta}_i)\|^2 + \frac{\sigma^2}{m}\right)\right) \\ &+ 2\|\nabla L(\boldsymbol{\theta}_i)\|\left(h\|H(\boldsymbol{\theta}_i)\| + \frac{\sigma}{\sqrt{m}}\right).\end{aligned}$$

Thus, the error bound for the diagonal Hessian estimation is:

$$\begin{aligned}&\|\hat{H}_i - H_i\| \\ =&O(h^2\|H(\boldsymbol{\theta}_i)\|^2 + \frac{\sigma^2}{m} + 2h\|\nabla L(\boldsymbol{\theta}_i)\|\|H(\boldsymbol{\theta}_i)\| \\ &+ \frac{2\|\nabla L(\boldsymbol{\theta}_i)\|\sigma}{\sqrt{m}}).\end{aligned}$$

Step 3. **Combination of the bounds**. Let $u_i = \mathrm{clip}\left((\hat{\boldsymbol{g}}_i\hat{\boldsymbol{g}}_i)^{-1}\hat{\boldsymbol{g}}_i, \lambda_i\right)$. By the definition of the clipping operation:

$$||u_i||_\infty \leq \lambda_i.$$

Thus:

$$||\boldsymbol{\theta}_i^+ - \boldsymbol{\theta}_i|| = \eta||u_i|| \leq \eta\lambda_i\sqrt{d_i}.$$

Define the function:

$$f(t) = L(\boldsymbol{\theta}_i + (1-t)u_i).$$

By Assumption 4.2, we know that:

$$f''(t) \leq 2f''(0) \quad \text{for all } t \in [0,1],$$

and hence:

$$\begin{aligned}f(1) =&f(0) + f'(0) + \int_0^1 \int_0^s f''(s)\, ds\, dt \\ \leq&f(0) + f'(0) + f''(0).\end{aligned}$$

The ZO estimate introduces noise ϵ in the estimated gradient:

$$\hat{\boldsymbol{g}}_i = \nabla L(\boldsymbol{\theta}_i) + \epsilon.$$

Thus:

$$f'(0) = -\eta\sum_{i=1}^{d} \min\left\{\lambda_i\,|\hat{\boldsymbol{g}}_i|,\, (\hat{\boldsymbol{g}}_i\hat{\boldsymbol{g}}_i)^{-1}|\hat{\boldsymbol{g}}_i|^2\right\}.$$

Using the bound for the error $||\epsilon|| \leq O\left(h\,||H(\boldsymbol{\theta}_i)|| + \frac{\sigma}{\sqrt{m}}\right)$, the noise affects the effective descent rate. Therefore, the new bound for $f'(0)$ is:

$$\begin{aligned}f'(0) \approx -\eta\sum_{i=1}^{d}\min\{&\lambda_i\,(|\nabla L(\boldsymbol{\theta}_i)| + |\epsilon|), \\ &(\hat{\boldsymbol{g}}_i\hat{\boldsymbol{g}}_i)^{-1}(|\nabla L(\boldsymbol{\theta}_i)| + |\epsilon|)^2\}.\end{aligned}$$

The Hessian is estimated using $\hat{\boldsymbol{g}}_i^2$. The noise in the diagonal Hessian estimate affects the curvature. Therefore, for the second derivative, we have:

$$f''(0) \leq \eta^2 \sum_{i=1}^{d} \min\left\{\lambda_i \, |\hat{\boldsymbol{g}}_i|, \, (\hat{\boldsymbol{g}}_i\hat{\boldsymbol{g}}_i)^{-1}|\hat{\boldsymbol{g}}_i|^2\right\}.$$

The noise in the Hessian (δ_H) affects the estimation, and thus the bound is affected as follows:

$$\begin{aligned} f''(0) \leq \eta^2 \sum_{i=1}^{d} \min\{&\lambda_i(|\nabla L(\boldsymbol{\theta}_i) + \epsilon|), \\ &(\hat{\boldsymbol{g}}_i\hat{\boldsymbol{g}}_i)^{-1}(|\nabla L(\boldsymbol{\theta}_i) + \epsilon|)^2\}. \end{aligned}$$

Combining these results, the descent bound is affected by both the gradient and Hessian noise. We obtain:

$$\begin{aligned} L(\boldsymbol{\theta}_i^+) - L(\boldsymbol{\theta}_i) \leq -(\eta - \eta^2) \sum_{i=1}^{d} \min\{&\lambda_i \, |\hat{\boldsymbol{g}}_i|, \\ &(\hat{\boldsymbol{g}}_i\hat{\boldsymbol{g}}_i)^{-1}|\hat{\boldsymbol{g}}_i|^2\} + C(\delta_g^2 + \delta_H^2), \end{aligned}$$

where C is a constant that depends on the properties of the function L. δ_g represents the bound on the gradient estimation noise $\delta_g = O\left(h \, ||H(\boldsymbol{\theta}_i)|| + \frac{\sigma}{\sqrt{m}}\right)$, and δ_H represents the bound on the Hessian estimation noise:

$$\begin{aligned} \delta_H = O(h^2||H(\boldsymbol{\theta}_i)||^2 + \frac{\sigma^2}{m} + \frac{2||\nabla L(\boldsymbol{\theta}_i)||\sigma}{\sqrt{m}} \\ + 2h \, ||\nabla L(\boldsymbol{\theta}_i)|| \, ||H(\boldsymbol{\theta}_i)||). \end{aligned}$$

□

Lemma 11 (Convergence Lemma). *For any $\lambda_i \leq \frac{R_i}{\sqrt{d_i}}$ and some $T_i \in \mathbb{N}$, if $L(\boldsymbol{\theta}_{T_i,i}) - \min L \leq \frac{\mu_i^2}{8}$, then for all $t \geq T_i$,*

1. *$\boldsymbol{\theta}_{t+1,i} = \boldsymbol{\theta}_{t,i} - \eta(\nabla^2_{\boldsymbol{\theta}_i} L(\boldsymbol{\theta}_{t,i}))^{-1}\nabla L(\boldsymbol{\theta}_{t,i})$,*
2. *$L(\boldsymbol{\theta}_{t,i}) - \min L \leq (1 - \eta(1 - \eta))^{t-T_i}(L(\boldsymbol{\theta}_{T_i,i}) - \min L)$.*

Proof. By Lemma 10, we have for all $t \geq T$, $(\boldsymbol{\theta}_{t,i}) - \min L \leq L(\boldsymbol{\theta}_{T,i}) - \min L \leq \frac{\mu^2}{8}$. Therefore, by Lemma 7, we have that $\|\nabla^2 L(\boldsymbol{\theta}_{t,i}) - \nabla L(\boldsymbol{\theta}_{t,i})\|_2 \leq \lambda_i$ for all $t \geq T$, which implies clipping will not happen.

For the second claim, by Lemmas 5 and 10, we have that

$$\begin{aligned} &L(\boldsymbol{\theta}_{t+1,i}) - L(\boldsymbol{\theta}_{t,i}) \\ &\leq -(\eta - \eta^2) \sum_{i=1}^{d} \sigma_i^{-1}|\mathbf{v}_i^T \nabla L(\boldsymbol{\theta}_{t,i})|^2, \end{aligned}$$

where v_i is the i-th row of matrix V from the eigendecomposition of $\nabla^2 L(\boldsymbol{\theta}_i)$. By further simplification,

$$\begin{aligned} &-(\eta - \eta^2)\nabla L(\boldsymbol{\theta}_{t,i})^T(\nabla^2 L(\boldsymbol{\theta}_{t,i}))^{-1}\nabla L(\boldsymbol{\theta}_{t,i}) \\ &\leq -\eta(1 - \eta)(L(\boldsymbol{\theta}_{t,i}) - \min L), \end{aligned}$$

thus, we conclude that the loss decreases at least geometrically by the factor $(1 - \eta(1 - \eta))$ each step after time T, thereby proving the convergence rate. □

Theorem 2. *Under Assumptions 1 and 2, let $\eta = \frac{1}{2}$ and $\lambda_i = \frac{R_i}{2\sqrt{d_i}}$. The update reaches a loss at most ϵ in*

$$\begin{aligned} T \leq \max_i \Bigg\{ &d_i \cdot \big(L(\boldsymbol{\theta}_{0,i}) - \min L\big) \\ &+ \ln\left(\frac{\mu_i R_i^2}{32 d_i \epsilon}\right)\Bigg\}. \end{aligned}$$

steps, where L is the loss function, $\boldsymbol{\theta}_{0,i}$ is the initial parameter vector for layer i.

Proof. By Lemma 10 (Descent Lemma), we have a guarantee on the descent rate per step for each layer i:

$$\begin{aligned} &L(\boldsymbol{\theta}_{t+1,i}) - L(\boldsymbol{\theta}_{t,i}) \\ &\leq -(\eta - \eta^2) \sum_{j=1}^{d_i} \min\left\{\lambda_i; \frac{1}{\sigma_{i,j}} \left|v_{i,j}^T \nabla L(\boldsymbol{\theta}_{t,i})\right|^2\right\}, \end{aligned}$$

where $\sigma_{i,j}$ is the j-th eigenvalue corresponding to the i-th layer, and $v_{i,j}$ is the corresponding eigenvector.

Applying this result, we estimate a decrease in the loss function per layer under the condition that the gradient norm for layer i is significantly larger than the error threshold ϵ. This phase continues until the loss reduction per step for each layer falls below a certain threshold, say when:

$$L(\boldsymbol{\theta}_{t,i}) - \min L \leq \frac{\mu_i^2}{8}.$$

Phase 2: Exponential Decay.

Once the loss for each layer is sufficiently reduced, Lemma 11 guides the convergence from this point:

$$\begin{aligned} &L(\boldsymbol{\theta}_{t,i}) - \min L \\ \leq&(1 - \eta(1 - \eta))^{t-T_i}(L(\boldsymbol{\theta}_{T_i,i}) - \min L), \end{aligned}$$

indicating an exponential decay in error for each layer. The factor $(1-\eta(1-\eta))$ represents the contraction per step, dependent on the learning rate η.

To calculate the total number of steps T_i for each layer, consider that:

$$\frac{\mu_i^2}{8} \approx \epsilon \Rightarrow T_i \approx \frac{\ln\left(\frac{L(\boldsymbol{\theta}_{0,i})-\min L}{\epsilon}\right)}{-\ln(1-\eta(1-\eta))}.$$

Simplifying the expression for $\eta = \frac{1}{2}$, we get:

$$T_i \approx 2\ln\left(\frac{L(\boldsymbol{\theta}_{0,i})-\min L}{\epsilon}\right),$$

since $\ln(1-\eta(1-\eta)) \approx -\eta(1-\eta)$ for small η.

Combining Phases 1 and 2.

For each layer, combining the estimates from Phase 1 and Phase 2, the total number of steps T_i needed to reach a loss of ϵ for layer i is given by:

$$T_i \leq d_i \cdot (L(\boldsymbol{\theta}_{0,i}) - \min L) + \ln\left(\frac{\mu_i R_i^2}{32 d_i \epsilon}\right),$$

Finally, to ensure convergence across all layers, we take the maximum over all layers:

$$T \leq \max_i \left\{ d_i \cdot (L(\boldsymbol{\theta}_{0,i}) - \min L) + \ln\left(\frac{\mu_i R_i^2}{32 d_i \epsilon}\right)\right\}.$$

This completes the proof by integrating the rapid initial decrease and the subsequent exponential decay for each layer.

□

This reflects an improved convergence rate due to the use of different λ_i values for different layers, reducing the dependency on the total dimension d into the dimension $\max_i d_i$.

G Memory Usage Analysis

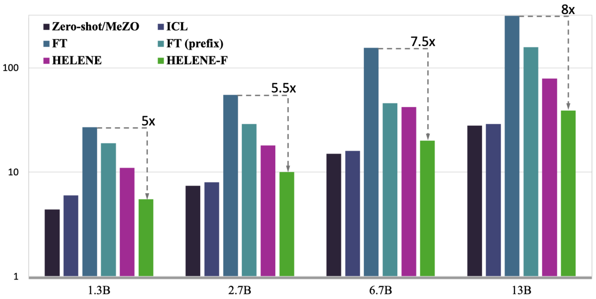


Figure 9: GPU memory consumption for finetuning different OPT models on the SST-2 dataset with a maximum sequence length of 1000 per example. More details can be found in Appendix 12.

Method	zero-shot/MeZO(FT)	HELENE	HELENE-F(FT)
1.3B	1xA100 (4GB)	1xA100 (11GB)	1xA100 (5.5GB)
2.7B	1xA100 (7GB)	1xA100 (18GB)	1xA100 (10GB)
6.7B	1xA100 (14GB)	1xA100 (42GB)	1xA100 (20GB)
13B	1xA100 (26GB)	1xA100 (79GB)	1xA100 (39GB)

Method	ICL	Adam(FT)
1.3B	1xA100 (6GB)	1xA100 (27GB)
2.7B	1xA100 (8GB)	1xA100 (55GB)
6.7B	1xA100 (16GB)	2xA100 (156GB)
13B	1xA100 (29GB)	4xA100 (316GB)

Table 12: Memory usage on the MultiRC (average tokens=400) dataset. Results of ICL and full-parameter tuning are from MeZO (Malladi et al., 2023).