OPTIMIZER SELECTION BASED ON FUNCTION-PROXY FIDELITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing highly nonconvex loss functions is central to modern machine learning, making optimizer selection both important and difficult. Many training algorithms can be interpreted as minimizing local surrogates of the loss, typically induced by Taylor approximations, but it is often unclear which surrogate is most appropriate for a given problem. At the same time, Hessian drift is poorly understood, and existing second-order methods typically rely on Tikhonov regularization to cope with concavity and departures from local quadratic structure. This regularization introduces additional tuning and can weaken the very second-order information these methods seek to exploit. We present a novel eigendecomposition-centric analysis of Hessian drift and use it to derive a new Hessian regularization method together with a meta-algorithm for optimizer selection. Experiments validate the effectiveness of our algorithms.

1 INTRODUCTION

Machine learning training consists of fitting models to target distributions such as protein structures, image labels, or text, yet no single training algorithm ("optimizer") is uniformly best across problems (Duchi et al., 2011b; Zeiler, 2012; Ruder, 2016; Xu et al., 2017; Mittal et al., 2019; Bera & Shrivastava, 2020; Zhou et al., 2020; Wadia et al., 2021; Schmidt et al., 2021; Mustapha et al., 2021; Sivan et al., 2024). This diversity of optimizers raises a fundamental question: how should one select an optimizer for a given problem? More ambitiously, because optimal update rules vary across parameters and training phases, an ideal framework would adaptively choose the best optimizer for each parameter online all throughout training. High-cost second-order updates are worthwhile only where their benefits amortize their computational overhead; conversely, relying strictly on first-order methods underuses valuable curvature information and increases iteration counts. Managing this balance is essential outside large-scale clusters, where training on commodity hardware, desktop GPUs, or edge platforms demands operating under rigid memory and throughput ceilings, while enabling private, low-latency learning. On these devices, adaptive optimization ensures compute and memory resources are deployed only where higher-order updates yield actionable gains. In federated and intermittently connected systems, reducing the number of synchronization rounds may also be more important than maximizing throughput on a single powerful accelerator.

A natural way to compare optimizers is empirical evaluation, and the literature contains extensive experimental studies across a wide range of applications (Xu et al., 2017; Schmidt et al., 2021). Theoretically, many convergence-rate guarantees have been proven for particular optimization methods. Yet these results often rely on heterogeneous assumptions and metrics, vary substantially in tightness, and rarely extend cleanly to new optimizers, making them difficult to use in practice. Moreover, general nonconvex guarantees often obscure curvature variation and are too loose to explain the benefits of second-order information, particularly in linear subspaces $\mathbb{S}$ in which the loss function $f|_{\mathbb{S}} : \mathbb{S} \to \mathbb{R}$ is locally concave [1]. Such directions are understudied despite evidence that near-optimal steps in these spaces can yield far larger gains than in locally convex subspaces (Alain et al., 2018; Ghorbani et al., 2019). Our bounds in Appendix C may help to overcome these challenges.

Our contributions

[1] Here, "locally concave" means that there exists a ball of radius $\delta > 0$ centered at x on which f is concave. Equivalently, its Hessian is negative semidefinite throughout that ball.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1. We introduce algorithm ELMO (Algorithm 1) and show that it is minmax-optimal[2] up to third order. We also show that this construction is a natural refinement of Newton's method, from which we derive a Hessian spectral regularization rule for general second-order optimizers that guarantees positive-definite preconditioners.
2. We derive AMOS (Alg. 2), which uses ELMO's limiting regimes to select between first- and second-order updates according to curvature variation.
3. We evaluate AMOS and ELMO's regularizer experimentally on edge-GPU training scenarios, and conduct an empirical study of the curvature drift throughout training.

Proofs appear in Appendix P; limitations in Appendix S, and code at https://github.com/ REDACTED .

2 BACKGROUND AND NOTATION

Assumption 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ denote the loss function. We assume that f is twice differentiable and has bounded Hessian, i.e., $\|\mathcal{H}\|_2 \leq L_\sigma$*

Notation 1. Let $\theta_{t+1}, \theta_t \in \mathbb{R}^n$ the parameter vectors of a pair of consecutive iterations of a given optimization algorithm. We mark $\Delta\theta_t \triangleq \theta_{t+1} - \theta_t$, $\nabla f(\theta_t)$ the gradient of f, and $\mathcal{H}(\theta_t)$ the Hessian of f at θ_t. We say an algorithm is a *k-order algorithm* if it requires oracle access to the first k derivatives of f.

Notation 2. We mark $(v_i(\theta_t), \lambda_i(\theta_t))_{i=0}^n$ an orthogonal eigendecomposition of $\mathcal{H}(\theta_t)$ (which exists due to the Hessian symmetry property). When the meaning is clear, we will sometimes shorten to v_i, λ_i. Since v_i and $-v_i$ are both equally viable eigenvectors, we eliminate ambiguity by assuming

$$\forall_{i\in[n]} : \nabla f(\theta_t)^\top v_i \leq 0 \tag{1}$$

with $\top$ indicating transposition.

Notation 3. For $\tau \in \mathbb{N}$ we mark $[\tau] = \{t \in \mathbb{N} : t \leq \tau\}$.

Notation 4. Let $A, B, D \in \mathbb{R}^{n\times n}$ s.t. $D = \mathit{diag}(d_1, d_2, \ldots, d_n)$ is diagonal, and let $\xi : \mathbb{R} \to \mathbb{R}$. We mark $\xi\left(A \cdot D \cdot A^\top\right) = A \cdot \mathit{diag}(\xi(d_1), \xi(d_2), \ldots, \xi(d_n)) \cdot A^\top$ the application of ξ to D. We write $A \succeq 0$ iff A is positive semi-definite, $A \succ 0$ if A is positive definite, $A \succeq B$ if $A - B \succeq 0$ (and likewise for $A \succ B$), and we mark $\lambda_{\min}(A), \lambda_{\max}(A)$ the minimal/maximal eigenvalue of A, respectively, and their ratio $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ the condition number of A.

Definition 1. We say that an optimization algorithm is a *quasi-Newton optimization algorithm* if its characteristic update rule may be expressed as:

$$\theta_{t+1} = \theta_t - \alpha_t \Phi_t \nabla f(\theta_t)$$

for $\Phi_t \in \mathbb{R}^{n\times n}, \Phi_t \succeq 0, \Phi_t^\top = \Phi_t, \alpha_t \in \mathbb{R}^+$. We call Φ_t in such algorithms the "preconditioner matrix". Throughout this paper, we will mark the point a convergent quasi-Newton algorithm converges to by θ^*.

This definition is motivated by Newton's method in convex optimization, in which $\Phi_t = (\mathcal{H}(\theta_t))^{-1}$ (see Nocedal & Wright (2006, Chapter 3)). A few successful quasi-Newton algorithms are reviewed in Appendix A. Since the vast majority of gradient-based optimizers can be expressed in this form (see Martens (2020) for examples), we restrict our analysis to this family throughout the paper.

2.1 RELATED WORK

Loss function bounds The loss value attained after t iterations is of primary practical interest, since it directly affects model quality. A common theoretical measure tied to this quantity is the

[2]For a given optimization algorithm, one can define an adversarial loss function that is consistent with all information observed at a given iteration while minimizing the descent achieved by that algorithm. We define the algorithm's regret as the gap between its achieved descent in this adversarial setting and the descent attained by the most adversary-robust algorithm, i.e., the one that maximizes descent under the same worst-case construction. An algorithm that minimizes this "regret" at each iteration is termed minmax-optimal.

objective-function suboptimality gap (OFSOG), defined as $f(\theta_T) - f(\theta^*)$. In this setting, the adaptive regularization with cubics (ARC) framework provides a notable second-order baseline. Cartis et al. (2012b) show that, in the convex regime, a variant of ARC attains OFSOG below ϵ in $\mathcal{O}\left(\epsilon^{-1}\right)$ iterations, and in the strongly convex regime in $\mathcal{O}\left(\log(\epsilon^{-1})\right)$ iterations. Similar guarantees are known for the nonlinear stepsize control family (Garmanjani, 2020), which generalizes both ARC and trust-region methods (Toint, 2013). More recently, Liu et al. (2024) establish OFSOG guarantees for Sophia, a stochastic second-order method based on diagonal Hessian approximations, with logarithmic dependence on ϵ^{-1} in the strictly convex regime. However, a broadly applicable bound of this type does not currently exist to the best of our knowledge.

First-order criticality bounds Most convergence analyses in the nonconvex regime focus on the complexity of reaching first-order criticality. For example, Wang et al. (2016) derive a bound for a broad class of optimizers under weak assumptions, showing convergence to an ϵ_g-stationary point within $\mathcal{O}\left(\kappa^{\frac{2}{1-\nu}}(\Phi_t) \cdot \epsilon_g^{-\frac{1}{1-\nu}}\right)$ iterations when the learning rate satisfies $\alpha_t = \mathcal{O}(t^{-\nu})$ for $\nu \in (0.5, 1)$. However, this bound is minimized when Φ_t minimizes $\kappa(\Phi_t)$, which is achieved by a scalar multiple of the identity, i.e., by gradient descent. Both theory and experiment (see appendix A) suggest that higher-order methods can outperform such first-order methods in our setting, indicating that first-order-criticality bounds of this kind may be too loose to capture the benefits of more informative preconditioning. By contrast, the bounds we develop in Appendix C capture those benefits more directly.

Multi-optimizer hybrids Several methods combine first- and second-order updates. Sivan et al. (2024) and Doikov et al. (2024) use Lanczos iterations for partial Hessian eigendecompositions, applying Newton steps in the dominant-eigenvalue subspace and first-order updates elsewhere. They enforce positive semidefiniteness by taking absolute eigenvalues or applying Tikhonov regularization, respectively. Both incur repeated-eigendecomposition costs and substantial memory overhead from storying the projection operators; moreover, Tikhonov regularization dilutes the Hessian signal, while absolute eigenvalues lack theoretical justification for the absolute Hessian values. Even so, in Appendix C we provide a new theoretical account for why such methods can nevertheless succeed.

A different line of work is represented by Ozkara et al. (2024), who optimize optimizer hyperparameters online by differentiating through the update rule itself. Their method gives a unified formulation of five first-order optimizers—Yogi (Zaheer et al., 2018), Adan (Xie et al., 2022), Adam (Kingma & Ba, 2014), AVGrad (Ozkara et al., 2024), and Lion (Chen et al., 2023)—and interpolates among them during training. However, this framework does not naturally extend to broader optimizer families, particularly second-order methods, and it introduces substantial memory overhead because it must maintain the state of all constituent optimizers as well as the intermediate activations required to backpropagate through the hyperparameter updates.

We refer the reader to appendix B for further discussion on convergence rates and the effect of the preconditioner (and correspondingly, of optimizer choice) on convergence rate.

3 A MINMAX HESSIAN LIPSCHITZ-AWARE OPTIMIZATION ALGORITHM

Optimization algorithm iterations are comprised of two parts: first, we gather information about the loss function and construct a local model of the loss function, and second - we step towards the minimum of this proxy. Accordingly, gradient descent and Newton's method use first- and second-order Taylor proxies, respectively; and although these do indicate descent directions, neither determines an optimal step in locally concave subspaces, where the corresponding polynomials have no minima. A third-order model is therefore needed to define a unique step in both nonconvex and nonquadratic regions, and to assess the second-order proxy's fidelity.

3.1 PER-ITERATION DESCENT REGRET

A recurring observation in the neural-network optimization literature is that the Hessian's leading eigenspaces tend to evolve slowly during training, both in their eigenvalues and in their eigenvectors; see, for example, Sagun et al. (2016); Alain et al. (2018); Gur-Ari et al. (2018); Ghorbani et al.

(2019); Liu et al. (2024); Sivan et al. (2024). A common formalization of this phenomenon is Hessian Lipschitz continuity in spectral norm (Nesterov & Polyak, 2006; O'Leary-Roseberry et al., 2019):

$$\exists_{L_H \in \mathbb{R}} \forall_{\theta,\varphi \in \mathbb{R}^n} : \|\mathcal{H}(\theta) - \mathcal{H}(\varphi)\|_2 \leq L_H \cdot \|\theta - \varphi\|_2 \tag{2}$$

This assumption summarizes the rate of Hessian variation using a single scalar constant. However, because the Hessian has $\frac{n^2}{2}$ degrees of freedom, its variation can be highly anisotropic. As a result, a single global constant may be overly coarse: satisfying equation 2 may require choosing L_H according to the most rapidly varying direction, which can make convergence guarantees loose and algorithms based on this quantity overly conservative. We therefore introduce a finer-grained assumption that tracks the evolution of the Hessian eigendecomposition itself.

Assumption 2 (Hessian Lipschitz-Continuity in each Eigenspace)**.** *For any $\theta, \varphi \in \mathbb{R}^n$, let (eigendecompositions)*

$$\mathcal{H}(\theta) = V \cdot \Lambda \cdot V^\top, \mathcal{H}(\varphi) = \tilde{V} \cdot \tilde{\Lambda} \cdot \tilde{V}^\top$$

with $V, \tilde{V} \in \mathbb{R}^{n \times n}$ orthogonal matrices and

$$\Lambda = \mathit{diag}\left(\lambda_i\right)_{i=1}^n, \tilde{\Lambda} = \mathit{diag}\left(\tilde{\lambda}_i\right)_{i=1}^n \in \mathbb{R}^{n \times n}$$

diagonal matrices, sorted s.t. $\forall_{i \in [n-1]} : \lambda_i \leq \lambda_{i+1}, \tilde{\lambda}_i \leq \tilde{\lambda}_{i+1}$. We assume:

- *Lipschitz-continuous eigenvalues*

$$\forall_{\theta \in \mathbb{R}^n} \exists_{\left(\bar{L}^i\right)_{i=1}^n \in (\mathbb{R}^+)^n} \forall_{\varphi \in \mathbb{R}^n} : \left|\lambda_i - \tilde{\lambda}_i\right| \leq \bar{L}^i \cdot \left|(\theta - \varphi)^\top v_i\right|$$

- *Lipschitz-continuous eigenvectors*

$$\forall_{\theta \in \mathbb{R}^n} \exists_{L_R \in \mathbb{R}^+} \forall_{\varphi \in \mathbb{R}^n} : \left\|V - \tilde{V}\right\|_2 \leq L_R \cdot \|\theta - \varphi\|_2$$

- *Bounded Lipschitz parameters*

$$\exists_{L_H \in \mathbb{R}} \forall_{\theta \in \mathbb{R}^n} \forall_{i \in [n]} : \max\left\{L_R, \bar{L}^i\right\} \leq L_H \text{ and } \bar{L}^i \geq L_H^{-1}$$

When θ is the t-th iterate θ_t of an optimization algorithm, we denote the corresponding Lipschitz parameters by L_t^i. Later, we show empirically that this refinement is meaningful in practice: these per-eigenspace parameters can differ by several orders of magnitude.

Remark 1. *Although Assumption 2 is more detailed than equation 2, it should not be viewed as substantially stronger. The constant L_H in the standard Hessian-Lipschitz assumption is, up to the additional eigenvector-control term L_R, essentially the same quantity as the constant L_H in equation 2. Indeed, by the equivalence between the matrix L_2-induced norm and the spectral norm (Roger A. Horn, 2013, Chapter 5.6, p. 346), the standard assumption bounds Hessian variation uniformly in every direction, and in particular dominates the largest per-eigenspace Lipschitz parameter. In this sense, the directional continuity captured by our assumption is already latent in the standard one; we simply make the corresponding eigenspace-wise Lipschitz parameters $\bar{L}^i$ explicit. The main new ingredient is the separate control of eigenvector variation through L_R. The converse direction is proven in Theorem P.4.*

Second-order methods typically gather the loss, its gradient, and optionally its Hessian at the current iterate. However, this local information does not uniquely determine the loss landscape: even within an arbitrarily small neighborhood of the current point, there are infinitely many functions that agree on value, gradient, and Hessian (while also satisfying Assumptions 1 and 2). From the optimizer's perspective, these functions are indistinguishable, so no second-order algorithm can guarantee that a given step reaches, even locally, the minimizer of the true loss.

We therefore adopt a minmax or regret-minimization viewpoint. At each iteration, we consider the worst-case loss function consistent with the information observed—that is, the adversarial function that yields the smallest possible descent for any given step (i.e. the pointwise maximum of all those infinite functions that are indistinguishable from the true loss function). We then choose the step that minimizes this adversarial upper envelope. In the lemma below, we also construct corresponding lower bounds, which together allow us to bracket the true loss from above and from below.

Notation 5. Let $\theta_t \in \mathbb{R}^n$, $v_i \in \mathbb{R}^n$ an eigenvector of $\mathcal{H}(\theta_t)$.

$$M_t^i(x) \triangleq \nabla f(\theta_t)^\top v_i \cdot x + \frac{v_i^\top \mathcal{H}(\theta_t) v_i}{2} \cdot x^2 + \frac{L_t^i}{6} \cdot |x|^3 \tag{3}$$

$$m_t^i(x) \triangleq \nabla f(\theta_t)^\top v_i \cdot x + \frac{v_i^\top \mathcal{H}(\theta_t) v_i}{2} \cdot x^2 - \frac{L_t^i}{6} \cdot |x|^3 \tag{4}$$

Lemma 3.1 (Eigenspace Descent Bounds)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function satisfying assumptions 1 and 2, and let $\theta_{t+1} \in \mathbb{R}^n$. Marking $\Delta\theta_t = \theta_{t+1} - \theta_t$, we have*

$$\exists_{\left(L_t^i\right)_{i=1}^n \in (\mathbb{R}^+)^n} : f(\theta_{t+1}) - f(\theta_t) \leq \sum_{i=1}^n M_t^i\left(\Delta\theta_t^\top v_i\right) \tag{5}$$

$$\exists_{\left(L_t^i\right)_{i=1}^n \in (\mathbb{R}^+)^n} : f(\theta_{t+1}) - f(\theta_t) \geq \sum_{i=1}^n m_t^i\left(\Delta\theta_t^\top v_i\right) \tag{6}$$

Remark 2. *In subspaces where the loss is locally concave (i.e., where the Hessian has a negative eigenvalue), the quadratic term in M_t^i from Lemma 3.1 lowers the adversarial loss model, revealing stronger guaranteed descent opportunities along these directions. This provides a new explanation for the empirical benefits of exploiting negative-curvature directions reported by Alain et al. (2018); Ghorbani et al. (2019).*

3.2 Exploiting these bounds for a minmax algorithm

Algorithm 1 Algorithm ELMO

Require: $\epsilon \in \mathbb{R}^+, \theta_0 \in \mathbb{R}^n$, EIGEN, LIPSCHITZ
  $t \leftarrow 0$
  while $f(\theta_t) - f(\theta^*) > \epsilon$ **do**
    $(\lambda_i, v_i)_{i=1}^n \leftarrow \texttt{EIGEN}(\mathcal{H}(\theta_t))$
    $\left(L_t^i\right)_{i=1}^n \leftarrow \texttt{LIPSCHITZ}(\theta_t, (v_i)_{i=1}^n)$
    $\left(\Delta\theta_t^i\right)_{i=1}^n \leftarrow \frac{\sqrt{\lambda_i^2 + 2L_t^i\left|\nabla f(\theta_t)^\top v_i\right|} - \lambda_i}{L_t^i}$
    $\theta_{t+1} \leftarrow \theta_t + \sum_{i=1}^n \Delta\theta_t^i \cdot v_i$
    $t \leftarrow t + 1$
  end while

Here we describe how to exploit the refined bounds that we obtain above to design a minmax-optimal algorithm named ELMO which is depicted in Alg. 1.

Due to the orthogonality of the Hessian eigenvectors v_i, the step components along different eigendirections decouple, allowing us to optimize the contribution of each eigenspace independently. We therefore minimize the RHS of equation 5 by minimizing each term separately, which amounts to choosing, in each eigenspace, the positive root of the derivative of the corresponding cubic model.

$$\Delta\theta_t^{*\top} v_i \triangleq \arg\min_{\Delta\theta_t} M_t^i\left(\Delta\theta_t^\top v_i\right) = \frac{\sqrt{\lambda_i^2 + 2L_t^i\left|\nabla f(\theta_t)^\top v_i\right|} - \lambda_i}{L_t^i} \tag{7}$$

The above naturally induces a second-order training algorithm that we name Eigenspace-Lipschitz Minmax Optimizer (ELMO); which we describe in Alg. 1. We present a theoretical analysis of ELMO in Appendix D, and in Appendix C we make use of ELMO as a minmax baseline to bound the sub-optimality of second-order algorithms, discovering a link to Sivan et al. (2024, FOSI). EIGEN and LIPSCHITZ denote an eigendecomposition subroutine and a Lipschitz-parameter oracle, respectively[3] However, because these components are computationally expensive, ELMO is an idealized reference algorithm whose exact oracle is used to derive practical regularization (sec. 3.2.1), optimizer regret analysis (sec. C), and optimizer-selection rules (sec. 4).

Importantly, because it does not assume convexity, equation 7—and hence ELMO—applies in both convex and concave regions of the domain space. In particular, ELMO exploits concavity by taking larger steps in concave subspaces, since the step size in its second term is proportional to $-\lambda_i$).

[3]We provide two approaches for computing this oracle: one inspired by ARC's Lipschitz-parameter estimation method (Appendix K.1.1) and one inspired by BFGS's (Broyden, 1970) Hessian estimation method (used in Algorithm AMOS below).

3.2.1 THE MINMAX PRECONDITIONER

Here we find algorithm ELMO's characteristic preconditioner. That is, we find Φ_t s.t. ELMO's step $\Delta\theta_t^*$ can be expressed as in definition 1. From it, we derive a general Hessian spectral regularization rule.

Lemma 3.2 (ELMO's preconditioner)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ satisfying assumptions 1 and 2. The preconditioner of the quasi-Newton algorithm that is equivalent to* ELMO *is*

$$\Phi_t^{\texttt{ELMO}} = \left(\frac{\mathcal{H}(\theta_t) + \sqrt{(\mathcal{H}(\theta_t))^2 + 2V \cdot D \cdot V^\top}}{2} \right)^{-1}$$

with

$$D = diag\left(L_t^i \cdot \left| \nabla f(\theta_t)^\top v_i \right| \right)_{i=1}^n \succeq 0$$

This preconditioner highlights the mechanistic similarity between our method and Newton's: whereas Newton's method uses the inverse Hessian directly (which may not be positive definite), our preconditioner is the inverse of a matrix obtained by averaging the Hessian with a positive-definite, regularized variant of the Hessian whose eigenvalues are all at least as large as the corresponding Hessian eigenvalues in magnitude. This construction ensures that the resulting preconditioner is positive semidefinite, with a level of regularization that adapts to the local rate of curvature change.

Newton's method may overshoot the local minimum In fact, Newton's method (and, by extension, many quasi-Newton methods that approximate it) can exhibit worst-case *decreases* in model quality even when the loss is globally convex, provided the curvature varies sufficiently rapidly (as measured by the Hessian Lipschitz constant). Substituting the Newton step into equation 3 and rearranging yields $\forall_{i \in [n] s.t. \lambda_i \geq 0} : M_t^i \left(\frac{\left| \nabla f(\theta_t)_t v_i \right|}{\lambda_i} \right) \geq 0$ for any step t and eigenspace i with

$$L_t^i \geq 3 \frac{\lambda_i^2}{\left| \nabla f(\theta_t)^\top v_i \right|} \tag{8}$$

Spectral regularization ELMO relies on an exact eigendecomposition of the Hessian, which can be prohibitively expensive for large models. However, the preceding insight suggests a practical way to enhance existing tractable second-order methods that already approximate the Hessian spectrum (see Appendix A). In particular, we can augment such methods with ELMO's regularization rule by replacing their eigenvalues in place with the regularized (nonnegative) values $\lambda_i \leftarrow \frac{\lambda_i + \sqrt{\lambda_i^2 + 2 \cdot L^i \cdot \left| \nabla f(\theta_t)^\top v_i \right|}}{2}$, making the preconditioner positive semi-definite even without Tikhonov damping. In the next section, we show how to estimate the Lipschitz parameters L^i within the AMOS algorithm.

4 DYNAMIC OPTIMIZER SELECTION

Here we exploit the insights from section 3.2.1 to develop a meta-optimizer which we name AMOS[4] (Alg. 2) that dynamically combines different optimizers throughout the training process in order to boost performance.

The limiting behavior of ELMO provides a direct criterion for deciding when second-order information is worth computing. In low-L_i^t eigenspaces, ELMO approaches the Newton update (Appendix G.2), so curvature information remains a reliable and useful description of the loss. In high-L_i^t eigenspaces, ELMO's guaranteed-descent step shrinks toward zero (Appendix I.1), indicating that the local quadratic model has lost predictive value. Moreover, the cost of obtaining this information is not reduced in the high-L_i^t regime. The practical consequence is therefore not to use a smaller ELMO step, but to avoid paying for a second-order update whose predicted benefit is small.

[4]AMOS: Adaptive Meta-algorithm for Optimizer Selection

The AMOS Meta-Optimizer AMOS approximates this rule by dynamically assigning each parameter block to either a specified second-order optimizer, OPT2, or a specified first-order optimizer, OPT1. Whenever OPT2 computes curvature information, AMOS updates its estimate of the corresponding Lipschitz parameter using an exponential moving average of the observed eigenvalue slope $L_t^i \approx \alpha_L L_{t-1}^i + (1-\alpha_L)\frac{|\lambda_i(\theta_t)-\lambda_i(\theta_{t-1})|}{\|\theta_t-\theta_{t-1}\|}$. Here, a single observed slope is used to estimate the computationally intractable upper bound defining the Lipschitz parameter; see Appendix O for justification. Because AMOS reuses the Hessian already computed by OPT2, estimating these slopes requires no additional Hessian evaluations. Between successive Hessian evaluations, blocks with $\widehat{L}_i^t > L_{\text{thresh}}$ are updated by OPT1, whereas blocks with $\widehat{L}_i^t \le L_{\text{thresh}}$ continue to use OPT2 (Algorithm 2, lines 7–10). Thus, AMOS is a computationally efficient approximation to the optimizer-selection behavior implied by ELMO's two limiting regimes. An analysis of AMOS's convergence rate may be found in Appendix Q.

Algorithm 2 Algorithm AMOS

Require: $\epsilon_1, \epsilon_2, L_{thresh}, \alpha_L \in \mathbb{R}^+, \theta_0 \in \mathbb{R}^n, I_{lpshz} \in 2^{\mathbb{N}}, \texttt{OPT}^1, \texttt{OPT}^2$

1: $t \leftarrow 0$
2: **repeat**
3: **if** $t \in I_{lpshz}$ **then**
4: $(\lambda_i^t)_{i=1}^n \leftarrow \texttt{EIGEN}(\texttt{OPT}^2.\texttt{hessian})$
5: $\left(L_t^i\right)_{i=1}^n \leftarrow \alpha_L \cdot L_{t-1}^i + (1-\alpha_L) \cdot \frac{\left|\lambda_i^t - \lambda_i^{t-1}\right|}{\epsilon_2 + \|\theta_t - \theta_{t-1}\|_2}$
6: **for** $i = 1..n$ **do**
7: **if** $L_t^i > L_{thresh}$ **then**
8: $\texttt{OPTIM}_\texttt{i} \leftarrow \texttt{OPT}^1$
9: **else**
10: $\texttt{OPTIM}_\texttt{i} \leftarrow \texttt{OPT}^2$
11: **end if**
12: **end for**
13: **end if**
14: $(\Delta\theta_t)_i = \texttt{OPTIM}_\texttt{i}.\texttt{step}()$
15: $(\theta_{t+1})_i \leftarrow (\theta_t)_i + (\Delta\theta_t)_i$
16: $t \leftarrow t+1$
17: **until** $f(\theta_{t-1}) - f(\theta_t) > \epsilon_1$

Remark 3. *AMOS does incur additional cost for Hessian eigendecomposition, but this overhead can be amortized by performing it infrequently, as is standard in practical second-order methods (e.g., every few hundred iterations in Sivan et al. (2024)); we note that one cannot perform Hessian eigendecomposition more frequently than the underlying second-order method computes the Hessian, due to AMOS's reuse of this same Hessian estimation. Moreover, most mainstream second-order optimizers already compute eigendecompositions (Sivan et al., 2024; Dauphin et al., 2014) (which AMOS can directly reuse), or assume a block-diagonal Kronecker-product Hessian (which admits fast and easily parallelizable eigendecomposition, and whose eigenvectors satisfy $v_{ij} = v_i^A \otimes v_j^B$ for $H = A \otimes B$) (Martens & Grosse, 2015, K-FAC), or assume a diagonal Hessian and thus **do not require an eigendecomposition** at all (Liu et al., 2024, Sophia).*

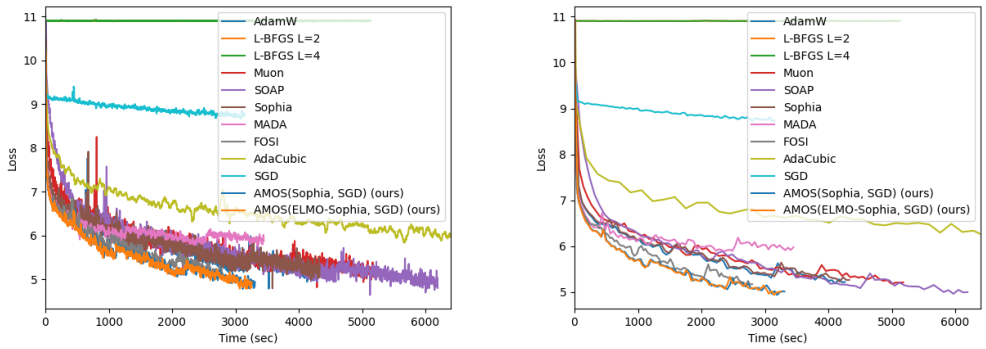
5 Experiments

We tested AMOS in the NanoGPT language modeling framework by Karpathy (2022) based on GPT-2.

- We trained a 6-layer transformer with 8-head multihead attention on the TinyShakespeare dataset (Karpathy, 2015) for 2,010 training iterations, and on the Fineweb-Edu (Penedo et al., 2024) dataset, OpenWebText dataset (Gokaslan et al., 2019), and PG19 dataset (Rae et al., 2019) for 16,010 iterations.
- We use AMOS to adaptively select between SGD and Sophia (chosen to maximize Remark 3) throughout training.
- Our baselines include LBFGS (Liu & Nocedal, 1989), Muon (Jordan et al., 2024), SOAP (Vyas et al., 2024), Sophia (Liu et al., 2024), AdaCubic (Tsingalis et al., 2026), MADA (Ozkara et al., 2024), FOSI (Sivan et al., 2024), SGD, and AdamW (Loshchilov & Hutter, 2017). SGD (stochastic gradient descent), Muon, MADA, and AdamW are first order optimizers, while LBFGS, SOAP, FOSI, and Sophia are second-order optimizers (although certain equivalence properties have been shown between Muon and Shampoo, a second-order optimizer). AdaCubic is a third-order optimizer.

All experiments were run on a single Nvidia RTX 2080 Ti GPU with 11 GB of memory, with the exception of FOSI and MADA which were unable to run on the RTX 2080 Ti due to insufficient

memory and so were run on a single (significantly more computationally powerful) Nvidia A6000 GPU instead. The memory constraints guided our selection of model structure and optimizers (for example, we must resort to using L-BFGS with a very small history size to fit in the memory - causing it to fail to converge!), and demonstrate AMOS's applicability in memory-constrained settings.



(a) Train loss: SGD-Sophia dynamic selection vs. pure optimizers

(b) Test loss: SGD-Sophia dynamic selection vs. pure optimizers

Figure 1: Training trajectory comparison. AMOS significantly outperforms alternatives

Full experimental settings and results are provided in Appendix L; for completeness, Figure 1 presents the FineWeb-Edu results. We observe that although pure SGD and Sophia each underperform against AdamW, hybridizing them within our framework enables them to overcome this gap and outperform the competing alternatives. In particular, AMOS-based hybrids, including variants using the ELMO-regularized Hessian introduced in Section 3.2.1, achieve the strongest overall performance across datasets. Interestingly, the AMOS-Sophia hybrids obtain lower overall training and test losses than pure Sophia after the same number of iterations despite diverting a portion of the model parameters to SGD, which we attribute to excessively large steps taken by pure Sophia (section 3.2.1).

Hyperparameter sensitivity experiments are presented in Appendix M.

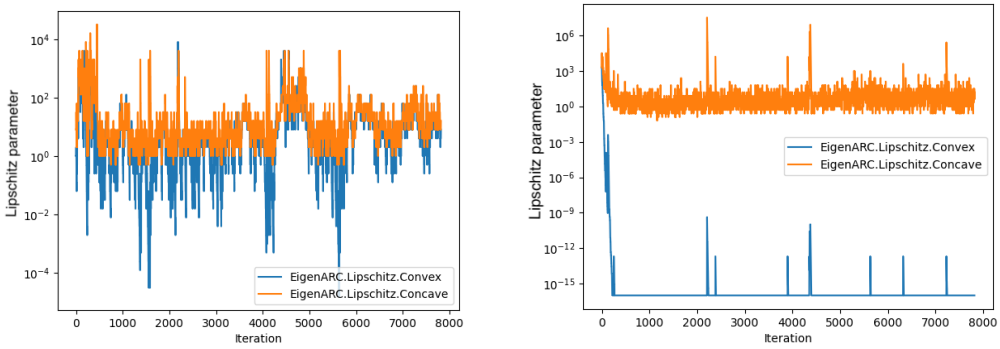
6 Lipschitz distribution

Notation 6. Taking $(\lambda_i)_{i=1}^n$ the eigenvalues of $\mathcal{H}(\theta)$ for some θ, note that since n is finite, there exist $L^+ \triangleq \max_i\{L^i : \lambda_i > 0\}, L^- \triangleq \max_i\{L^i : \lambda_i \leq 0\}$.

Previous works using the Hessian Lipschitz continuity assumption (e.g. ARC (Nesterov & Polyak, 2006) and its variants, O'Leary-Roseberry et al. (2019)) assume a single Lipschitz parameter for all eigenspaces. Although a finite number n of eigenspaces ensures that such a Lipschitz parameter exists (the maximal Lipschitz parameter), they fail to account for the distribution of these Lipschitz parameters over the eigenspaces. We claim that these parameters vary widely both over the eigenspaces and over the course of training, so that a single constant value fails to capture this structure; in this section we provide evidence for this claim. A discussion of sources of interest in this distribution is deferred to Appendix H.

The first source of evidence for our claim is from existing literature on the subject; we defer a discussion of this to appendix J. To test our claim directly, we modify an ARC implementation (Simpson & Wang, 2023) to compute the steps called for by ELMO at each point reached by a quasi-Newton algorithm, restricted to the subspace spanned by the eigenvectors corresponding to the single most positive and single most negative eigenvalues of each Hessian, and to use distinct Lipschitz parameters for each (L^+ and L^-, respectively). Due to the computational difficulty of computing Lipschitz parameters precisely, we use these estimated Lipschitz parameter values as an estimate for L^+, L^-. We note the crudeness of these adaptive measurements, merely adapting to keep $\frac{f(\theta_t)-f(\theta_{t+1})}{\left|\sum_{i=1}^n M_t^i\right|}$ within a given range with a restriction to powers of 2; nevertheless, the point is made.

A detailing of our experiment settings is given in appendix K as well as the full set of our experiment results, however we present two experiments in figure 2 for completeness. Our experiments show that as expected, $L^+ \ll L^-$, and the gap widens exponentially as training progresses in all cases except the autoencoders. Since we saw in section 3.2.1 that small convex Lipschitz parameters imply effective second-order optimization, this justifies common practice as noted by, e.g. O'Leary-Roseberry et al. (2019), of requiring the preconditioner to be an increasingly better approximation of the inverse Hessian (by increasing the strictness of the inverse Hessian approximation algorithm's stopping condition) as training progresses. Our findings also support Gur-Ari et al. (2018)'s observation that neural network optimization succeeds despite its nonconvexity due to the gradient residing primarily in convex subspaces, by demonstrating the stability of these convex subspaces.



(a) Training autoencoder on CIFAR10

(b) Training Resnet on CIFAR10 with cross-entropy loss

Figure 2: Comparisons of convex-subspace Lipschitz parameters to concave-subspace Lipschitz parameters. *Logarithmic scale*

Several factors seem to impact the size (by orders of magnitude) of the convex Lipschitz parameters, and they appear to be correlated with an intuitive sense of the difficulty of the setting being trained.

- The convex Lipschitz parameters are many orders of magnitude greater in the autoencoder task than in the classification task. We ascribe this gap to the more difficult task of learning a generative representation of the data instead of merely a discriminative representation of it (see Ng (2012, Chapter 4) for a discussion on generative vs. discriminative models).
- The convex Lipschitz parameters are reduced approximately 100x in the image classification task by adding residual connections. It is well known that residual connections reduce training difficulty (Li et al., 2018).
- The convex Lipschitz parameters are approximately 100x smaller when training ResNet to perform classification of natural images instead of Gaussian noise with random labels. We ascribe this to greater difficulty involved in discriminating noise, which requires partial memorization of the training set.

The Lipschitz parameters' indication of Newton's applicability suggests that the above observations may be applicable as a rule of thumb for optimizer selection; this is validated in experiments in Appendix F.

7 Conclusion

In this work, we developed and analyzed `ELMO`, a Hessian-eigenspace, Lipschitz-aware minmax optimization algorithm. Its update has a simple limiting interpretation: it becomes Newton-like when curvature is stable and its guaranteed-descent step vanishes when curvature varies rapidly. This interpretation leads directly to `AMOS`, which selectively uses second-order updates in the former regime and first-order updates in the latter, using inexpensive estimates of the Lipschitz parameters. In addition, ELMO yields a curvature-drift-aware spectral regularization rule that can be incorporated into practical second-order optimizers. We also investigated the distribution of Hessian Lipschitz parameters and found that it correlates with task difficulty.

REFERENCES

Natalie Abreu, Nikhil Vyas, Sham M. Kakade, and Depen Morwani. The potential of second-order optimization for llms: A study with full gauss-newton. *ArXiv*, abs/2510.09378, 2025. URL https://api.semanticscholar.org/CorpusID:282056128.

Naman Agarwal, Brian Bullins, Xinyi Chen, Elad Hazan, Karan Singh, Cyril Zhang, and Yi Zhang. Efficient full-matrix adaptive regularization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 102–110. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/agarwal19b.html.

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Learning rate grafting: Transferability of optimizer tuning, 2022. URL https://openreview.net/forum?id=FpKgG31Z_i9.

Guillaume Alain, Nicolas Le Roux, and Pierre-Antoine Manzagol. Negative eigenvalues of the hessian in deep neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1iiddyDG.

Zeyuan Allen-Zhu. Katyusha: the first direct acceleration of stochastic gradient methods. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2016. URL https://api.semanticscholar.org/CorpusID:51918825.

Shun-ichi Amari. Natural Gradient Works Efficiently in Learning. *Neural Computation*, 10(2): 251–276, 02 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL https://doi.org/10.1162/089976698300017746.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Second order optimization made practical. *CoRR*, abs/2002.09018, 2020. URL https://arxiv.org/abs/2002.09018.

Yossi Arjevani, Yair Carmon, John C. Duchi, Dylan J. Foster, Nathan Srebro, and Blake E. Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199:165–214, 2019. URL https://api.semanticscholar.org/CorpusID:208637439.

Somenath Bera and Vimal Shrivastava. Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing*, 41:2664–2683, 04 2020. doi: 10.1080/01431161.2019.1694725.

Dimitri P. Bertsekas. Incremental least squares methods and the extended kalman filter. *SIAM J. Optim.*, 6(3):807–822, 1996. doi: 10.1137/S1052623494268522. URL https://doi.org/10.1137/S1052623494268522.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 557–565. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/botev17a.html.

Léon Bottou. Stochastic learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch (eds.), *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, pp. 146–168. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-28650-9. doi: 10.1007/978-3-540-28650-9_7. URL https://doi.org/10.1007/978-3-540-28650-9_7.

Léon Bottou and Yann Lecun. On-line learning for very large datasets. *J. Applied Stochastic Models in Business and Industry*, 01 2004.

C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 03 1970. ISSN 0272-4960. doi: 10.1093/imamat/6.1.76. URL https://doi.org/10.1093/imamat/6.1.76.

C. Cartis, N.I.M. Gould, and Ph.L. Toint. Complexity bounds for second-order optimality in unconstrained optimization. *Journal of Complexity*, 28(1):93–108, 2012a. ISSN 0885-064X. doi: https://doi.org/10.1016/j.jco.2011.06.001. URL https://www.sciencedirect.com/science/article/pii/S0885064X11000537.

Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011a. doi: 10.1007/s10107-009-0286-5. URL https://doi.org/10.1007/s10107-009-0286-5.

Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function- and derivative-evaluation complexity. *Mathematical Programming*, 130(2):295–319, 2011b. doi: 10.1007/s10107-009-0337-y. URL https://doi.org/10.1007/s10107-009-0337-y.

Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. Evaluation complexity of adaptive cubic regularization methods for convex unconstrained optimization. *Optimization Methods and Software*, 27:197 – 219, 2012b. URL https://api.semanticscholar.org/CorpusID:16647191.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. *ArXiv*, abs/2302.06675, 2023. URL https://api.semanticscholar.org/CorpusID:256846990.

Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, PA, USA, 2000.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2933–2941, Cambridge, MA, USA, 2014. MIT Press.

Alexandre D'efossez, Léon Bottou, Francis R. Bach, and Nicolas Usunier. A simple convergence proof of adam and adagrad. *Trans. Mach. Learn. Res.*, 2022, 2020. URL https://api.semanticscholar.org/CorpusID:225213299.

Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982. doi: 10.1137/0719025. URL https://doi.org/10.1137/0719025.

J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall Civil Engineering and Engineering Mechanics Se. Prentice-Hall, 1983. ISBN 9780136272168. URL https://books.google.co.il/books?id=4fFQAAAAMAAJ.

Tian Ding, Dawei Li, and Ruoyu Sun. Sub-optimal local minima exist for almost all over-parameterized neural networks. *ArXiv*, abs/1911.01413, 2019. URL https://api.semanticscholar.org/CorpusID:207870322.

Nikita Doikov, Sebastian U. Stich, and Martin Jaggi. Spectral preconditioning for gradient methods on graded non-convex functions. In *International Conference on Machine Learning*, 2024. URL https://api.semanticscholar.org/CorpusID:267523218.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011a. URL http://jmlr.org/papers/v12/duchi11a.html.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011b. URL https://api.semanticscholar.org/CorpusID:538820.

S C Eisenstat and H F Walker. Choosing the forcing terms in an inexact newton method. *SIAM Journal on Scientific Computing*, 17(1), 1 1996. doi: 10.1137/0917003. URL https://www.osti.gov/biblio/218521.

William R. Esposito and Christodoulos A. Floudas. Gauss–newton method: Least squares, relation to newton's methodgauss–newton method: Least squares, relation to newton's method. In Christodoulos A. Floudas and Panos M. Pardalos (eds.), *Encyclopedia of Optimization*, pp. 733–738. Springer US, Boston, MA, 2001. ISBN 978-0-306-48332-5. doi: 10.1007/0-306-48332-7_151. URL https://doi.org/10.1007/0-306-48332-7_151.

Reuben Feinman. Pytorch-minimize: a library for numerical optimization with autograd, 2021. URL https://github.com/rfeinman/pytorch-minimize.

Dan Garber, Elad Hazan, Chi Jin, Kakade Sham, Cameron Musco, Praneeth Netrapalli, and Aaron Sidford. Faster eigenvector computation via shift-and-invert preconditioning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2626–2634, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/garber16.html.

R. Garmanjani. A note on the worst-case complexity of nonlinear stepsize control methods for convex smooth unconstrained optimization. *Optimization*, 71:1–11, 10 2020. doi: 10.1080/02331934.2020.1830088.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points — online stochastic gradient for tensor decomposition. In Peter Grünwald, Elad Hazan, and Satyen Kale (eds.), *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pp. 797–842, Paris, France, 03–06 Jul 2015. PMLR. URL https://proceedings.mlr.press/v40/Ge15.html.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.*, 23:2341–2368, 2013. URL https://api.semanticscholar.org/CorpusID:14112046.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2232–2241. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19b.html.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2386–2396. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/192fc044e74dffea144f9ac5dc9f3395-Paper.pdf.

I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14 (1):107–114, 1952. ISSN 00359246. URL http://www.jstor.org/stable/2984087.

Andreas Griewank. The modification of newton's method for unconstrained optimization by bounding cubic terms. Technical report, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, United Kingdom, 1981.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/gupta18a.html.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *ArXiv*, abs/1812.04754, 2018. URL https://api.semanticscholar.org/CorpusID:54480858.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015. URL https://api.semanticscholar.org/CorpusID:206594692.

Maarten Jansen and Gerda Claeskens. Cramér–rao inequality. In Miodrag Lovric (ed.), *International Encyclopedia of Statistical Science*, pp. 322–323. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_197. URL https://doi.org/10.1007/978-3-642-04898-2_197.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1724–1732. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/jin17a.html.

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

Andrej Karpathy. char-rnn. https://github.com/karpathy/char-rnn, 2015.

Andrej Karpathy. NanoGPT. https://github.com/karpathy/nanoGPT, 2022.

Kenji Kawaguchi. Deep learning without poor local minima. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/f2fc990265c712c49d51a18a32b39f0c-Paper.pdf.

Kenji Kawaguchi and Yoshua Bengio. Depth with nonlinearity creates no bad local minima in resnets. *Neural Networks*, 118:167–174, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.06.009. URL https://www.sciencedirect.com/science/article/pii/S0893608019301820.

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

Yann LeCun. *PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)*. PhD thesis, Université Pierre et Marie Curie, Paris, France, 1987. URL https://api.semanticscholar.org/CorpusID:151887454.

Kfir Yehuda Levy. The power of normalization: Faster evasion of saddle points. *ArXiv*, abs/1611.04831, 2016. URL https://api.semanticscholar.org/CorpusID:16706102.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf.

Lin and Segel. *Mathematics Applied to Deterministic Problems in the Natural Sciences*. SIAM, Philadelphia, 1988. ISBN 978-0-89871-229-2. doi: 10.1137/1.9781611971347. URL https://doi.org/10.1137/1.9781611971347.

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989. URL https://api.semanticscholar.org/CorpusID:5681609.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3xHDeA8Noi.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017. URL https://api.semanticscholar.org/CorpusID:3312944.

Kenneth L. Manders and Leonard Adleman. Np-complete decision problems for binary quadratics. *Journal of Computer and System Sciences*, 16(2):168–184, 1978. ISSN 0022-0000. doi: https://doi.org/10.1016/0022-0000(78)90044-2. URL https://www.sciencedirect.com/science/article/pii/0022000078900442.

James Martens. Deep learning via hessian-free optimization. In Johannes Fürnkranz and Thorsten Joachims (eds.), *ICML*, pp. 735–742. Omnipress, 2010. URL http://dblp.uni-trier.de/db/conf/icml/icml2010.html#Martens10.

James Martens. New insights and perspectives on the natural gradient method. *J. Mach. Learn. Res.*, 21(1), jan 2020. ISSN 1532-4435.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/martens15.html.

Harshal Mittal, Kartikey Pandey, and Yash Kant. Iclr reproducibility challenge report (padam : Closing the generalization gap of adaptive gradient methods in training deep neural networks). *ArXiv*, abs/1901.09517, 2019. URL https://api.semanticscholar.org/CorpusID:249647677.

Aatila Mustapha, Lachgar Mohamed, and Kartit Ali. Comparative study of optimization techniques in deep learning: Application in the ophthalmology field. *Journal of Physics: Conference Series*, 1743, 2021. URL https://api.semanticscholar.org/CorpusID:234179873.

Yurii Nesterov and B. T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006. doi: 10.1007/s10107-006-0706-8. URL https://doi.org/10.1007/s10107-006-0706-8.

Andrew Ng. Cs229 lecture notes - supervised learning. Available at https://cs229.stanford.edu/lectures-spring2022/main_notes.pdf, 2012.

Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJgXsjA5tQ.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2e edition, 2006.

Thomas O'Leary-Roseberry, Nick Alger, and Omar Ghattas. Inexact newton methods for stochastic non-convex optimization with applications to neural network training. *arXiv: Optimization and Control*, 2019. URL https://api.semanticscholar.org/CorpusID:155100112.

P.J. Olver and C. Shakiban. *Applied Linear Algebra*. Prentice Hall, 2006. ISBN 9780131473829. URL https://books.google.co.il/books?id=D2tyQgAACAAJ.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen,

Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

Kaan Ozkara, Can Karakus, Parameswaran Raman, Mingyi Hong, Shoham Sabach, Branislav Kveton, and Volkan Cevher. Mada: Meta-adaptive optimizers through hyper-gradient descent. *ArXiv*, abs/2401.08893, 2024. URL https://api.semanticscholar.org/CorpusID:267028245.

Panos M. Pardalos and Stephen A. Vavasis. Quadratic programming with one negative eigenvalue is np-hard. *Journal of Global Optimization*, 1(1):15–22, 1991. doi: 10.1007/BF00120662. URL https://doi.org/10.1007/BF00120662.

Dylan Patel and Gerald Wong. Gpt-4 architecture, infrastructure, training dataset, costs, vision, moe. https://www.semianalysis.com/p/gpt-4-architecture-infrastructure, 2023. Accessed: 2024-05-01.

Barak A. Pearlmutter. Fast Exact Multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 01 1994. ISSN 0899-7667. doi: 10.1162/neco.1994.6.1.147. URL https://doi.org/10.1162/neco.1994.6.1.147.

Karl Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London Series I*, 58:240–242, January 1895.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *ArXiv*, abs/2406.17557, 2024. URL https://api.semanticscholar.org/CorpusID:270711474.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1911.05507.

Charles R. Johnson Roger A. Horn. *Matrix Analysis, 2nd edition*, chapter 5.6. Cambridge University Press, 2013.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747, 2016. URL https://api.semanticscholar.org/CorpusID:17485266.

Levent Sagun, Léon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv: Learning*, 2016. URL https://api.semanticscholar.org/CorpusID:35723845.

Robin M Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley - benchmarking deep learning optimizers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9367–9376. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/schmidt21a.html.

Nicol Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14:1723–38, 08 2002. doi: 10.1162/08997660260028683.

Cooper Simpson and Jaden Wang. PyTorch-ARC. github.com/RS-Coop/PyTorch-ARC, 2023. Adaptive Regularization with Cubics (ARC) optimizer for PyTorch.

Hadar Sivan, Moshe Gabel, and Assaf Schuster. FOSI: Hybrid first and second order optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NvbeD9Ttkx.

Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *ArXiv*, abs/1605.08361, 2016. URL https://api.semanticscholar.org/CorpusID:3029264.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Philippe L. Toint. Nonlinear stepsize control, trust regions and regularizations for unconstrained optimization. *Optimization Methods and Software*, 28(1):82–95, 2013. doi: 10.1080/10556788.2011.610458. URL https://doi.org/10.1080/10556788.2011.610458.

J.F. Traub. *Iterative Methods for the Solution of Equations*. AMS Chelsea Publishing Series. Chelsea, 1982. ISBN 9780828403122. URL https://books.google.co.il/books?id=se3YdgFgz4YC.

Ioannis Tsingalis, Constantine Kotropoulos, and Corentin Briat. Adacubic: An adaptive cubic regularization optimizer for deep learning. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://api.semanticscholar.org/CorpusID:286515439.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. Soap: Improving and stabilizing shampoo using adam. *ArXiv*, abs/2409.11321, 2024. URL https://api.semanticscholar.org/CorpusID:272694107.

Neha Wadia, Daniel Duckworth, Samuel S Schoenholz, Ethan Dyer, and Jascha Sohl-Dickstein. Whitening and second order optimization both make information in the dataset unusable during training, and can reduce or prevent generalization. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10617–10629. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/wadia21a.html.

Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27, 07 2016. doi: 10.1137/15M1053141.

Rachel Ward, Xiaoxia Wu, and Leon Bottou. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6677–6686. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ward19a.html.

Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:9508–9520, 2022. URL https://api.semanticscholar.org/CorpusID:251564316.

Peng Xu, Farbod Roosta-Khorasan, and Michael Mahoney. Second-order optimization for non-convex machine learning: An empirical study. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, 08 2017.

Peng Xu, Fred Roosta, and Michael W. Mahoney. Newton-type methods for non-convex optimization under inexact hessian information. *Mathematical Programming*, 184(1):35–70, 2020. doi: 10.1007/s10107-019-01405-z. URL https://doi.org/10.1007/s10107-019-01405-z.

Robert M. Young. 75.9 euler's constant. *The Mathematical Gazette*, 75(472):187–190, 1991. ISSN 00255572. URL http://www.jstor.org/stable/3620251.

Xiao-Hu Yu and Guo-An Chen. On the local minima free condition of backpropagation learning. *IEEE Transactions on Neural Networks*, 6(5):1300–1303, 1995. doi: 10.1109/72.410380.

Manzil Zaheer, Sashank J. Reddi, Devendra Singh Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *Neural Information Processing Systems*, 2018. URL https://api.semanticscholar.org/CorpusID:54063095.

Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012. URL https://api.semanticscholar.org/CorpusID:7365802.

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, and Weinan E. Towards theoretically understanding why sgd generalizes better than adam in deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21285–21296. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf.

Milija Županski. A preconditioning algorithm for large-scale minimization problems. *Tellus A: Dynamic Meteorology and Oceanography*, Jan 1993. doi: 10.3402/tellusa.v45i5.15048.

A Quasi-Newton Challenges and Proposed Solutions

Some of the challenges involved in training neural networks include:

- Because the models and data often have very complex structures, obtaining precisely optimal parameters is often computationally prohibitive. As a result, one must satisfy oneself with a small degree of suboptimality in the model's parameters, chosen to be small enough to satisfy one's needs while not exhausting the computational capacity at hand.
- Since many model architectures (e.g. artificial neural networks) have very complex structures, the loss function is generally non-convex as a function of the model's parameters. This makes finding the globally optimal choice of parameters an NP-hard problem (Pardalos & Vavasis, 1991; Manders & Adleman, 1978). As a result, one must satisfy oneself with merely a local minimum of the loss function (that is, a point at which the norm of the gradient w.r.t. the model parameters is zero, and the function is locally convex, or equivalently, the Hessian is positive semi-definite). This is often considered sufficient (see Soudry & Carmon (2016); Kawaguchi & Bengio (2019); Kawaguchi (2016); Nguyen et al. (2019)), however this does not apply to saddle points and local maxima (points at which the gradient norm is zero but the Hessian is not positive definite). Although some work has been done on trying to eliminate this problem by eliminating local- but not global- minima via overparameterization (Yu & Chen, 1995), further work (Ding et al., 2019) has shown that this does not scale to deep neural networks.
- As mentioned previously, there is no single universally optimal optimizer, even among existing optimizers.

As a result, sophisticated optimizers are necessary to contend with different neural network training scenarios. Restricting ourselves to quasi-newton optimization algorithms, scenarios with $n \gg 0$ (a common theme in machine learning, where n may be in the millions, billions, or even trillions, as GPT4 (OpenAI et al., 2024) is rumored to have. See Patel & Wong (2023)) are that computing and inverting the Hessian (with respective complexities $\mathcal{O}\left(n^2\right), \mathcal{O}\left(n^3\right)$) may be computationally prohibitive. Also, one must ensure that

$$\Phi_t \succeq 0 \tag{9}$$

to ensure that $\theta_{t+1} - \theta_t$ is a descent direction of f. This is because $-\nabla f\left(\theta_t\right)$ is a descent direction of f, which implies that for all $v \in \mathbb{R}^n$, $-\alpha \cdot \nabla f\left(\theta_t\right)^\top v \cdot v^\top$ is a descent direction for $\alpha > 0$ and an ascent direction for $\alpha < 0$. However, if (λ_i, v_i) is an eigenvalue-eigenvector pair of Φ_t with $\lambda_i < 0$ then $-v_i^\top \cdot \Phi_t \nabla f\left(\theta_t\right) \cdot v_i = -\lambda_i \nabla f\left(\theta_t\right)^\top \cdot v_i \cdot v_i$ which is an ascent direction, and then a better preconditioner could immediately be obtained by taking $\tilde{\Phi}_t$ with eigenpairs $\left(\tilde{\lambda_j}, \tilde{v_j}\right), \tilde{v_j} = v_j, \tilde{\lambda_j} = \begin{cases} \lambda_j & j \neq i \\ 0 & j = i \end{cases}$ to prevent an ascent in the subspace (a.k.a. eigenspace) *span*(v_i).

Three common ways to contend with these challenges are:

- **The Hessian-Free approach** Making use of Pearlmutter (1994) to compute Hessian-vector products without explicit computation of the Hessian, one uses conjugate-gradient (Olver & Shakiban, 2006) iterations to compute progressively finer approximations to $\left(\mathcal{H}\left(\theta_t\right)\right)^{-1} \cdot \nabla f\left(\theta_t\right)$, stopping when one reaches a dimension with negative curvature. See, for instance, Martens (2010).
- **The Lanczos eigendecomposition approach** Making use of Lanczos iterations (Olver & Shakiban, 2006), one decomposes the Hessian into its eigendecomposition, and explicitly edits its eigenvalues. See, for instance, Dauphin et al. (2014); Sivan et al. (2024).
- **The Gauss-Newton approach** Using the generalized Gauss-Newton approximation to the Hessian (Esposito & Floudas, 2001; Schraudolph, 2002), one can obtain a matrix which has the following good properties:
  - Well approximated by a Kronecker product (sparse representation), which allows one to represent it and multiply by it very cheaply
  - Positive semi-definite
  - Can be computed with only a first-order loss function gradient oracle

- Well approximates the true loss Hessian, when the second derivative of the model or the residual loss ($f(\theta_t) - f(\theta^*)$) is insignificant next to the generalized Gauss Newton.

Some examples of this approach include Agarwal et al. (2019); Botev et al. (2017); Gupta et al. (2018); Martens & Grosse (2015); Goldfarb et al. (2020); Anil et al. (2020). Of particular note are examples that make diagonal approximations to the Gauss-Newton, as noted by Martens (2020), that are most often viewed as first-order methods, such as Adagrad (Duchi et al., 2011a), RMSProp (Tieleman & Hinton, 2012), and Adam (Kingma & Ba, 2014). As noted by Martens (2020), due to the strong connection between the generalized Gauss-Newton and the Fischer Information matrix (when the loss function is cross-entropy loss (Good, 1952)), one can achieve certain theoretical benefits when using such methods, such as Fischer efficiency; see Amari (1998) for instance, which views θ_t as an unbiased estimator of θ^* of f, and uses the Cramer-Rao inequality (Jansen & Claeskens, 2011) to lower-bound the minimal number of iterations required to minimize the variance of said estimator as a function of the Fischer Information due to the number of samples consumed by each iteration.

See Nocedal & Wright (2006, Chapters 3.3,3.4) for further discussion of these approaches.

In order for a minimization problem to be well-defined, one must assume that f is lower-bounded. We can infer from this that any subset of the domain space in which f is concave must be a bounded set (because nonconstant concave functions with unbounded domains are not lower-bounded); this means that the second-order Taylor approximation of the function must have a bounded neighborhood in which it approximates the function well. Additionally, even in subsets of the domain space in which f is convex, the neighborhood in which the second-order Taylor approximation of the loss function well-approximates the true loss function may be bounded. To address this, two common approaches been proposed in the literature, namely:

- **The Trust Regions Approach**, which explicitly maintains a radius of the neighborhood in which the second-order Taylor polynomial is a good approximation of the function, and bounds the step size to that radius. See Conn et al. (2000), Nocedal & Wright (2006, Chapter 4).
- **The Cubic Regularization Approach**, which assumes that Hessian is Lipschitz continuous (using the L2 vector-induced matrix norm to measure distances between Hessians), and as such can upper bound the distance between two points of the function using a third-order polynomial (discussed below, see Lemma 4.1.14 from Dennis & Schnabel (1983)). See Nesterov & Polyak (2006) for an algorithm based on this approach that adaptively estimates the Hessian-Lipschitz parameter.

B Other convergence rate measures

Convergence rates to first-order criticality As noted above, most works give first-order criticality convergence bounds, meaning that they bound the number of iterations to reduce the gradient norm below a given threshold. D'efossez et al. (2020) give such convergence rate bounds (requiring t iterations, for t s.t. $\frac{\sqrt{t}}{\log(t)} = \Omega\left(\epsilon_g^{-1}\right)$) for Adam and Adagrad, and Ward et al. (2019) give such convergence rate bounds (at $\mathcal{O}\left(\epsilon_g^{-1}\right)$) for gradient descent with Adagrad-grafted step-sizes (see Agarwal et al. (2022) for a discussion on learning rate grafting).

Convergence rates to second-order criticality A few go further in bounding the number of steps required to achieve second-order criticality (a point satisfying $\|\nabla f(\theta_t)\|_2 < \epsilon_g, -\lambda_{\min}(\mathcal{H}(\theta_t)) < \epsilon_H$). For instance, Nesterov & Polyak (2006); Cartis et al. (2011b); Xu et al. (2020) provide such bounds (at $\mathcal{O}\left(\max(\epsilon_g, \epsilon_H)^{-3}\right)$) on variants of the ARC algorithm, and Levy (2016); Jin et al. (2017); Ge et al. (2015) provide such bounds for varieties of SGD. This is of great importance since as noted, local minima are generally considered sufficiently optimal while local maxima/saddle points are not, despite being impossible to distinguish with only first-order criticality information. To the best of our knowledge, however, no such bounds exist in the general setting, nor do they exist for the vast majority of existing optimization algorithms.

Objective function sub-optimality gap bounds Bottou (2004) partition the optimization process into the initial "search phase", in which the optimizer searches for an approximately convex region in which a local minimum resides, and the later "final phase", in which the optimizer converges to this local minimum. In the machine learning literature, many common loss functions are "empirical risk functions" - that is, loss functions which can be written as a sum of terms, each of which is a function of only a single sample from the data distribution. When this sum ranges over a very large number of samples, a common approach to estimating it is to perform a Monte Carlo approximation, summing over only a small subset of the terms; this approach is known as the "minibatch approach". Amari (1998) then note that when using this approach, θ_t may be seen as a statistical estimator for θ^*. Working in the "final phase" (and thus assuming convexity), and adopting the estimator approach to θ_t taken by Amari (1998); Bottou & Lecun (2004) give a convergence rate bound for this estimator's variance parameterized by the first- and second-order derivatives at θ^*, assuming only that $\lim_{t\to\infty} \Phi_t = \mathcal{H}^{-1}(\theta^*)$. Martens (2020) takes these convergence rates and plugs them into a Taylor approximation of $f(\theta_t)$ to obtain the asymptotic OFSOG, given by $f(\theta_T) - f(\theta^*) = \frac{n}{2T} + o\left(\frac{1}{T}\right)$. By contrast, our bound does not require the loss function to be convex, making it applicable to the "search phase" as well.

Convergence rate dependence on preconditioner quality One possible quality metric for Φ_t is given by $\eta_t \triangleq \left\| (I - \mathcal{H}(\theta_t) \cdot \Phi_t) \cdot \frac{\nabla f(\theta_t)}{\|\nabla f(\theta_t)\|} \right\|_2$. In the convex regime, Nocedal & Wright (2006, Chapter 7.1) assume $\sup_t(\eta_t) < 1$ and prove that first-order criticality may be reached within $\mathcal{O}\left(\frac{\log \epsilon}{\log \frac{1+\sup_t(\eta_t)}{2}}\right)$ iterations. Adding an assumption of Lipschitz-continuity of the Hessian, they prove quadratic convergence to first-order criticality. O'Leary-Roseberry et al. (2019), in contrast, do not assume convexity but provide a bound on the parameter gap $\|\theta_t - \theta^*\|_2$ for η_t satisfying the Eisenstat-Walker (Eisenstat & Walker, 1996; Dembo et al., 1982) condition $\eta_t \leq \|\nabla f(\theta_t)\|_2$ on a Tikhonov-regularized Hessian. Like Wang et al. (2016), however, here too the constant in their bound is inversely proportional to $\zeta - \lambda_{\min}(\mathcal{H}(\theta_t))$ with ζ the Tikhonov regularization constant, thus is minimized by taking $\zeta \to \infty$, eliminating all second-order information and reverting to simple gradient descent. As before, this implies looseness due to the empirical success of making use of second-order methods.

C MINMAX-SUBOPTIMALITY OF ARBITRATRY QUASI-NEWTON ALGORITHMS

Algorithm ELMO is regret-optimal among first- and second-order methods in the sense that its model of the loss function is a generalization of quasi-Newton methods' and Gradient Descent's models (Appendix G), so it is natural to use it as a baseline against which to measure other optimizers' regret.

Notation 7. For a given algorithm with step $\Delta\theta_t$ at iteration t, mark $\Delta\Delta^i\theta_t = \Delta\theta_t^\top v_i - \Delta\theta_t^{*\top} v_i$ the step's distance from ELMO's step. Since $\Delta\Delta^i\theta_t$ is a function of the algorithm chosen (through $\Delta\theta_t$), it is a function of that algorithm's defining preconditioner: $\Delta\Delta^i\theta_t = \Delta\Delta^i\theta_t(\Phi_t)$

This may be understood intuitively by noting that $\sum_{i=1}^n \Delta\Delta^i\theta_t$ is the L_1 norm of the difference between steps, as measured over the Hessian eigenbasis.

Different datasets imply different units for the parameters of neural networks, however (e.g. a neural network whose training dataset measures distances in miles will have differently scaled parameters than one whose distances are measured in kilometers, and different optimizer step sizes accordingly), so to maintain consistency across a variety of settings, we turn to dimensional analysis (Lin & Segel, 1988) to eliminate this effect on $\Delta\Delta^i\theta_t$. For simplicity, we restrict our discussion to the descent of the loss function's restriction to a given eigenspace *span* (v_i).

Notation 8. Mark $\Delta\Delta^i\theta'_t = \frac{\Delta\Delta^i\theta_t}{\Delta\theta_t^{*\top} v_i}$ the step's distance from ELMO's step relative to ELMO's step.

In fact, this optimizer metric generalizes several existing metrics, see Appendix E for details.

Theorem C.1 (Worst-case descent rate for arbitrary optimizers)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ a twice-differentiable function satisfying assumptions 1 and 2, and let $\Delta\theta_t$ satisfy $M_t^i\left(\Delta\theta_t^\top v_i\right) \leq 0$. Then*

$$\left|M_t^i\left(\Delta\theta_t^\top v_i\right)\right| = \Theta\left(1 + \left|\Delta\Delta^i\theta'_t\right|^2\right) \cdot \left|M_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right|$$

$$\left|m_t^i\left(\Delta\theta_t^\top v_i\right)\right| = \Theta\left(1+\left|\Delta\Delta^i\theta_t'\right|^p\right)\cdot\left|m_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right| \tag{10}$$

$$\textit{with } p=\begin{cases}2 & \lambda_i>0 \textit{ and } \frac{\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}=0\\ 1 & else\end{cases}.$$

In other words, an arbitrary algorithm's descent regret (the ratio between the algorithm's descent in the adversarial setting and the optimal descent in the adversarial setting) is proportional to a polynomial of $\Delta\Delta^i\theta_t'$.

We hope future work will apply Theorem C.1 to bound algorithmic regret alongside proving convergence guarantees. Its relatively weak and transparent assumptions make it easy to use in practice, while its ability to quantify the benefit of exploiting second-order information makes it especially well suited to the analysis of second-order methods. To control the Hessian Lipschitz parameters appearing in the bound, we recommend hybridizing the target second-order optimizer with SGD using AMOS, and then showing that the resulting hybrid is minmax near-optimal on the parameters assigned to the second-order update. This automatically bounds the relevant Lipschitz parameters to the (constant) range $[0, L_{thresh}]$ for arbitrary $L_{thresh} > 0$. If the Lipschitz threshold is chosen sufficiently small to allow that term to be neglected (since we saw that the Lipschitz parameters are indeed small in convex subspaces), demonstrating regret near-optimality in a convex subspace is then reduced to proving that the optimizer's algorithm approximates the inverse Hessian well. For concave subspaces, the eigenvalues are negative, so the Hessian terms cancel each other out (when approximating the squareroot in $\Delta\Delta^i\theta_t$ with a linear approximation via Corollary P.1.1)

$$\Delta\theta_t^{*\top} v_i = \frac{\lambda_i+\sqrt{\lambda_i^2+2\cdot L^i\cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}}{2} \tag{11}$$

$$\approx \frac{\lambda_i+|\lambda_i|+\frac{2\cdot L^i\cdot\left|\nabla f(\theta_t)^\top v_i\right|}{2\cdot|\lambda_i|}}{2} \tag{12}$$

$$= \frac{L^i}{2}\cdot\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{|\lambda_i|} \tag{13}$$

Notice that the second term of the RHS of equation 13 is just the update step of Saddle-Free Newton (Dauphin et al., 2014) (after excluding their Tikohonov regularization), so when the Lipschitz threshold is selected sufficiently small as to regard the Lipschitz parameter as constant, proving regret near-optimality is reduced to demonstrating similarity of the proposed step to Saddle-Free Newton's step (Dauphin et al., 2014). This also gives a novel theoretical explanation for Dauphin et al. (2014)'s experimental success (and FOSI's (Sivan et al., 2024) as well, which uses a very similar update rule to Saddle-Free Newton, differing primarily by the update steps taken in the directions not addressed by the second-order update) despite their step being originally unjustified theoretically in concave subspaces of the loss function.

D Diminishing returns of additional ELMO iterations

An important factor in deciding how much computational effort to invest each optimization iteration is the ratio between the additional computational burden and the corresponding model quality improvement (as measured by the loss function) obtained by that iteration. To that end, we demonstrate that an upper bound on algorithm ELMO's performance has quickly diminishing rewards for additional iterations. Counter-intuitively, this is a good thing - it means that as long as the algorithm converges to an acceptable minimum point, just a few iterations are likely to be necessary in practice - since any more than that will not have much of an effect on the model's quality anyway. We recall that ELMO is the optimal regret-minimizing algorithm. This optimality yields an interesting fundamental implication for guaranteed descent: **ELMO's per-iteration minmax optimality implies that its $O(1/t)$ worst-case descent certificate is an upper bound on the descent that any method can guarantee from the same local information. Hence, along the ELMO reference trajectory, the cumulative certifiable descent of any method is at most $O(\log T)$.**

Theorem D.1 (Diminishing regret gains)**.** *Let f be a function with Lipschitz-continuous Hessian. After t iterations, algorithm* `ELMO` *satisfies*

$$f(\theta_0) - f(\theta_t) = \mathcal{O}(\log t) \tag{14}$$

Although the above theorem gives only an upper bound on the model's performance, we demonstrate that it is actually within a constant multiplicative factor of the algorithm's lower bound.

Lemma D.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ satisfying assumptions 1 and 2. Algorithm* `ELMO` *satisfies*

$$\left|m_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right| \leq 5\left|M_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right|$$

E GENERALIZATION OF PREVIOUS QUASI-NEWTON PRECONDITIONER QUALITY METRICS

Since most prevalent quasi-Newton algorithms apply a principled approach only to the convex subspaces of the loss function domain space and even then only when the curvature shift is negligible, we examine the special case of our metric when $\lambda_i > 0$ (when the loss function is convex over the domain subspace under examination) and show that our quality metric for quasi-Newton algorithm steps generalizes previous metrics. When $\lambda_i > 0$, we have

$$\left|\Delta\Delta^i\theta_t'\right| = \left|1 - \frac{\nabla f(\theta_t)^\top}{\nabla f(\theta_t)^\top v_i} \cdot \left(\alpha_t \Phi_t \mathcal{H}(\theta_t)\right) \cdot v_i \cdot S\right| \tag{15}$$

with

$$S = \frac{\sqrt{1 + 2L_t^i \cdot \frac{\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}} + 1}{2}$$

Županski (1993) introduce the "Effective Hessian" (a.k.a. the "Preconditioned Hessian") as $\mathcal{I}_t = \alpha_t \Phi_t \mathcal{H}(\theta_t)$, whose condition number used as a quality metric for preconditioners; ideally, $\kappa(\mathcal{I}_t) < \kappa(\mathcal{H}(\theta_t))$. The Effective Hessian may be plainly seen in equation 15.

Mark $r_t \triangleq (I - \mathcal{H}(\theta_t) \cdot \Phi_t) \cdot \frac{\nabla f(\theta_t)}{\nabla f(\theta_t)^\top v_i}$; this is the 1-dimensional version of the quality metric η_t for Φ_t used by Nocedal & Wright (2006, Chapter 7.1) and mentioned in appendix B (now redefined by projecting $\nabla f(\theta_t)$ onto the i-th eigenspace instead of taking its full norm). When $L^+ \approx 0$ (i.e. when the loss function curvature shift is negligible), equation 15 simplifies to

$$\left|\Delta\Delta^i\theta_t'\right| \approx \left|r_t^\top \cdot v_i\right|$$

F A QUALITY PREDICTOR FOR NEWTON'S METHOD

An important challenge is finding the best balance between per-iteration computational burden and expected loss function descent. We set out to provide such a metric due to equation 15 by showing that the expected descent in a given eigenspace is an approximately monotonically decreasing function of the corresponding Lipschitz parameter.

Figure 3 shows an example of this phenomenon by plotting the quasi-Newton superiority (how much better a quasi-Newton method will work than a first-order method, defined as $\left(f(\theta_t) - f\left(\theta_{t+1}^{Newton}\right)\right) - \left(f(\theta_t) - f\left(\theta_{t+1}^{SGD}\right)\right)$) against the convex Lipschitz parameter rank. Here too we represent the full spectrum of convex Lipschitz parameters with the single Lipschitz parameter representing the eigenspace with the greatest eigenvalue; nevertheless, a qualitative inverse correlation is clear. Pearson correlation coefficient values (Pearson, 1895) are shown in table 1, as well as p-values of a test of the null hypothesis that the distributions underlying the samples are uncorrelated and normally distributed. The Scipy manual writes:

> The p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Pearson correlation at least as extreme as the one computed from these datasets.

Table 1: Pearson r inverse correlation between quasi-Newton superiority and Lipschitz parameter

DATASET	PEARSON r	p-VALUE
CIFAR10	-0.245341	10^{-107}
FAKEDATA	-0.026608	0.031120
MNIST	-0.368788	10^{-300}

Here too, the detailing of our experiment settings is given in appendix K, as well as further detailing on figure 3.

Since the Lipschitz parameters are approximately locally constant throughout training as shown in the previous section, this reverse correlation may be used to help practitioners decide how much computational burden is worth putting into each iteration, given that even an exact Newton step may not be significantly superior to first-order methods when the curvature drift (as measured by Lipschitz parameters) is significantly large; hyperparameter optimization algorithm selection may then follow accordingly. We present experiments validating this selection method in appendix N. Alternatively, practitioners may choose to use ARC steps instead of first-order methods, when the Lipschitz parameter is significantly large.

G COMPARISON OF ELMO TO SELECT RELATED METHODS

ELMO is strikingly similar to Cauchy's method (not to be confused with Cauchy's Steepest Descent method (Nocedal & Wright, 2006, Chapter 4.1)) and Newton's method mentioned above. In this section, we note the similarity between them, and the sources of the differences between them.

G.1 COMPARISON TO CAUCHY'S METHOD

Cauchy's method (Traub, 1982) is very reminiscent of ELMO:

$$
\begin{aligned}
\left(\theta_{t+1}-\theta_t\right)_{cauchy}^{\top}\cdot v_i &\triangleq -\frac{2}{1+\sqrt{1-2\frac{L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}{\lambda_i^2\left(\theta_t\right)}}}\cdot\frac{\nabla f\left(\theta_t\right)^{\top}v_i}{\lambda_i\left(\theta_t\right)}\\
&= -\frac{2}{1+\frac{1}{|\lambda_i(\theta_t)|}\cdot\sqrt{\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}}\frac{\nabla f\left(\theta_t\right)^{\top}v_i}{\lambda_i\left(\theta_t\right)}\\
&= \frac{-2\nabla f\left(\theta_t\right)^{\top}v_i}{\lambda_i\left(\theta_t\right)+\frac{\lambda_i(\theta_t)}{|\lambda_i(\theta_t)|}\cdot\sqrt{\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}}\\
&= \frac{1}{L_t^i}\cdot\frac{-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}{\lambda_i\left(\theta_t\right)+\frac{\lambda_i(\theta_t)}{|\lambda_i(\theta_t)|}\cdot\sqrt{\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}}\\
&= \frac{1}{L_t^i}\cdot\frac{\left(\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i\right)-\lambda_i^2\left(\theta_t\right)}{\lambda_i\left(\theta_t\right)+\frac{\lambda_i(\theta_t)}{|\lambda_i(\theta_t)|}\cdot\sqrt{\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}}\\
&= \frac{\sqrt{\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}-\sqrt{\lambda_i^2\left(\theta_t\right)}}{L_t^i}\cdot\frac{|\lambda_i\left(\theta_t\right)|}{\lambda_i\left(\theta_t\right)}\\
&= \frac{\sqrt{\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}-\frac{|\lambda_i(\theta_t)|}{\lambda_i(\theta_t)}\cdot\lambda_i\left(\theta_t\right)}{L_t^i}\cdot\frac{|\lambda_i\left(\theta_t\right)|}{\lambda_i\left(\theta_t\right)}\\
&= \frac{\frac{|\lambda_i(\theta_t)|}{\lambda_i(\theta_t)}\cdot\sqrt{\lambda_i^2\left(\theta_t\right)-2L_t^i\cdot\nabla f\left(\theta_t\right)^{\top}v_i}-\lambda_i\left(\theta_t\right)}{L_t^i}
\end{aligned}
$$

The difference between our minimization step and their step is the sign on the squareroot. The difference stems from removing the absolute value in equation 3's 3rd-order term and taking the negative root of its derivative, due to the difference in goals: we attempt to minimize the function, leading us to select the positive step. They attempt to find the function's critical points, leading them to select the negative step.

G.2 Comparison to Newton's method

Unlike Cauchy's method, Newton's method (in optimization) makes a second-order approximation to the function's gradient. This is equivalent to the Hessian being constant, which is equivalent to $L_H = 0$. Indeed, taking the limit of equation 7 when $L_H \to 0^+$, we recover Newton's method:

$$\begin{aligned}\lim_{L_H \to 0^+} \Delta\theta_t^{*\top} v_i &= \lim_{L_H \to 0^+} -\frac{2\nabla f(\theta_t)^\top v_i}{\sqrt{\lambda_i^2(\theta_t) - 2L_t^i \cdot \nabla f(\theta_t)^\top v_i} + \lambda_i(\theta_t)} \\ &= \begin{cases} -\frac{\nabla f(\theta_t)^\top v_i}{\lambda_i} & \lambda_i > 0 \\ \infty & \lambda_i < 0 \end{cases}\end{aligned} \tag{16}$$

H Relevance of the Hessian-Lipschitz distribution

One source of interest in the Hessian-Lipschitz distribution is for optimization algorithms (e.g. ARC) that make use of these parameters for the loss function modelling stage of each iteration. This may reduce computational complexity by reducing the number of parameters one must compute at each iteration, however appendix I shows that poorly estimating the Lipschitz parameters can have a detrimental effect on an algorithm's descent rate (thereby increasing the number of iterations the algorithm will require to converge).

Another source of interest in these parameters' distribution is in explaining the effectiveness of second-order quasi-Newton algorithms that implicitly assume the Lipschitz parameters are insignificant (i.e. very close to zero), since their model of the loss function is a quadratic Taylor polynomial (i.e. no curvature shift); this may be seen from equation 15 which shows optimality of Newton's method only when $\lambda_i > 0$ and $L^+ = 0$. As seen in section 6, the Lipschitz parameters are not generally small by any means, however the Lipschitz parameters of the subspaces in which they work (the convex subspaces - see the implementation of Sivan et al. (2024), for instance, which applies Newton's method only on subspaces with significantly convex subspaces) are in fact small in certain settings.

I Descent rate dependence on Hessian-Lipschitz parameter

As noted by Griewank (1981), the Hessian-Lipschitz parameter (in our case, the respective constants of each eigenspace) may be computationally difficult to obtain precisely, leading some optimization algorithms to estimate it approximately instead of computing it precisely (e.g. ARC). In order to balance the computational burden of computing it to a high degree of exactitude with the degradation of an algorithm's descent rate that comes with poor estimations, we study the effects of the Hessian-Lipschitz parameter on $M_t^i\left(\Delta\theta_t^\top v_i\right)$.

I.1 Lipschitz Robustness

To address the descent rate's robustness to overly conservative L_t^i, we consider the case when $L_t^i \to \infty$.

Theorem I.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ satisfying assumptions 1 and 2. Then when $L_t^i \to \infty$, we have*

$$M_t^i\left(\Delta\theta_t^{*\top} v_i\right) = \Theta\left(\frac{1}{\sqrt{L_t^i}}\right)$$

Proof.

$$\lim_{L_t^i \to \infty} \frac{M_t^i\left(\Delta\theta_t^{*\top} v_i\right)}{\frac{-12\left(-\nabla f(\theta_t)^\top v_i\right)^{1.5} - 36\left(-\nabla f(\theta_t)^\top v_i\right) + \sqrt{2}\sqrt{-\nabla f(\theta_t)^\top v_i} + 3\sqrt{2}}{\frac{3}{-\nabla f(\theta_t)^\top v_i} - 18\sqrt{2}} \cdot \frac{1}{\sqrt{L_t^i}}}$$

$$= \lim_{L_t^i \to \infty} -\frac{1}{\sqrt{\frac{\lambda_i^2(\theta_t)}{L_t^i\left(-2\cdot\nabla f(\theta_t)^\top v_i\right)} + 1} + \frac{\lambda_i(\theta_t)}{\sqrt{L_t^i}\sqrt{-2\cdot\nabla f(\theta_t)^\top v_i}}} \cdot \frac{6\sqrt{-\nabla f\left(\theta_t\right)^\top v_i}}{6\sqrt{-\nabla f\left(\theta_t\right)^\top v_i} - 2\cdot\nabla f\left(\theta_t\right)^\top v_i}$$

$$- \frac{2\cdot\nabla f\left(\theta_t\right)^\top v_i}{6\sqrt{-\nabla f\left(\theta_t\right)^\top v_i} - 2\cdot\nabla f\left(\theta_t\right)^\top v_i} \cdot \left(\sqrt{\frac{\lambda_i^2\left(\theta_t\right)}{L_t^i\left(-2\cdot\nabla f\left(\theta_t\right)^\top v_i\right)} + 1} - \frac{\lambda_i\left(\theta_t\right)}{\sqrt{L_t^i}\sqrt{-2\cdot\nabla f\left(\theta_t\right)^\top v_i}}\right)^3$$

$$+ \frac{1}{\sqrt{L_t^i}} \cdot \frac{1 + 6\sqrt{2}\nabla f\left(\theta_t\right)^\top v_i}{2\left(6\sqrt{-\nabla f\left(\theta_t\right)^\top v_i} - 2\cdot\nabla f\left(\theta_t\right)^\top v_i\right)} \frac{\lambda_i\left(\theta_t\right)\left(\frac{-2\cdot\nabla f(\theta_t)^\top v_i}{\sqrt{\frac{\lambda_i^2(\theta_t)}{L_t^i} - 2\cdot\nabla f(\theta_t)^\top v_i} + \frac{\lambda_i(\theta_t)}{\sqrt{L_t^i}}}\right)^2}{\left(\frac{1}{6} + \sqrt{2}\nabla f\left(\theta_t\right)^\top v_i\right)\sqrt{-2\cdot\nabla f\left(\theta_t\right)^\top v_i}}$$

$$= 1$$

□

I.2 Benefit of Lipschitz tightness

To see how minimizing L_t^i as much as possible benefits the bound, we consider the case when $L_t^i \to 0^+$.

Theorem I.2 (Benefit of Lipschitz tightness: concave subspaces)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ satisfying assumptions 1 and 2. If $\lambda_i(\theta_t) \leq 0$ then when $L_t^i \to 0^+$, we have*

$$M_t^i\left(\Delta\theta_t^{*\top} v_i\right) = \Theta\left(-\frac{1}{L_t^{i,2}}\right)$$

Proof.

$$\lim_{L_t^i \to 0^+} \frac{M_t^i\left(\Delta\theta_t^{*\top} v_i\right)}{\frac{2}{3}\frac{\lambda_i^3(\theta_t)}{L_t^{i,2}}}$$

$$= \lim_{L_t^i \to 0^+} \left(3\left(\frac{\sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{(-\lambda_i(\theta_t))^2}} + 1}{2}\right)^2 - 2\left(\frac{\sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{(-\lambda_i(\theta_t))^2}} + 1}{2}\right)^3\right)$$

$$+ L_t^i \cdot \frac{3}{2\lambda_i^3\left(\theta_t\right)} \cdot \left(\sqrt{\lambda_i^2\left(\theta_t\right) - 2L_t^i \cdot \nabla f\left(\theta_t\right)^\top v_i} - \lambda_i\left(\theta_t\right)\right) \cdot \nabla f\left(\theta_t\right)^\top v_i$$

$$= 1$$

□

Theorem I.3 (Benefit of Lipschitz tightness: convex subspaces)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ satisfying assumptions 1 and 2. If $\lambda_i(\theta_t) > 0$ then when $L_t^i \to 0^+$, we have*

$$M_t^i\left(\Delta\theta_t^{*\top} v_i\right) - M_t^i\left(\lim_{L_t^i \to 0^+} \Delta\theta_t^{*\top} v_i\right) = \Theta\left(L_t^i\right)$$

Proof. We begin by noting that by equation 16, $\lim_{L_t^i \to 0^+} \Delta\theta_t^{*\top} v_i = \frac{\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i(\theta_t)}$. Plugging this into M_t^i:

$$
\begin{aligned}
&\lim_{L_t^i \to 0^+} \frac{M_t^i\left(\Delta\theta_t^{*\top} v_i\right) - M_t^i\left(\lim_{L_t^i\to 0^+} \Delta\theta_t^{*\top} v_i\right)}{\frac{L_t^i}{\frac{1}{2}\lambda_i(\theta_t)\cdot\frac{\left(\nabla f(\theta_t)^\top v_i\right)^3}{\lambda_i^4(\theta_t)} - \frac{\left(\nabla f(\theta_t)^\top v_i\right)^3}{2\lambda_i^2(\theta_t)}}} \\
&= \lim_{L_t^i \to 0^+} -\frac{\left(\nabla f\left(\theta_t\right)^\top v_i\right)^3}{2\lambda_i^2\left(\theta_t\right)} \cdot \left(\frac{4}{\left(\sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{\lambda_i(\theta_t)}} + 1\right)^2}\right) \\
&+ \frac{1}{2}\lambda_i\left(\theta_t\right) \cdot \frac{\left(\nabla f\left(\theta_t\right)^\top v_i\right)^3}{\lambda_i^4\left(\theta_t\right)} \cdot \left(\frac{\frac{2\lambda_i(\theta_t)}{\sqrt{\lambda_i^2(\theta_t) - 2L_t^i\cdot\nabla f(\theta_t)^\top v_i} + \lambda_i(\theta_t)} + 1}{2}\right) \\
&\cdot \left(\frac{2\lambda_i^2\left(\theta_t\right)}{\lambda_i\left(\theta_t\right)\sqrt{\lambda_i^2\left(\theta_t\right) - 2L_t^i \cdot \nabla f\left(\theta_t\right)^\top v_i} + \lambda_i^2\left(\theta_t\right)}\right)\left(\frac{2}{1 + \sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{\lambda_i^2(\theta_t)}}}\right) \\
&- \frac{L_t^i}{4} \cdot \frac{\left(\nabla f\left(\theta_t\right)^\top v_i\right)^4}{\lambda_i^5\left(\theta_t\right)}\left(\frac{2}{\sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{\lambda_i^2(\theta_t)}} + 1}\right)\left(\frac{2}{\sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{\lambda_i^2(\theta_t)}} + 1}\right) \\
&\cdot \left(\frac{1 + \frac{2}{\sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{\lambda_i^2(\theta_t)}} + 1} + \frac{4}{\left(\sqrt{1 - 2L_t^i \cdot \frac{\nabla f(\theta_t)^\top v_i}{\lambda_i^2(\theta_t)}} + 1\right)^2}}{3}\right) \\
&= 1
\end{aligned}
$$

□

J Qualitative evidence from previous work on Lipschitz parameters

Experiments by Alain et al. (2018); Sagun et al. (2016); Ghorbani et al. (2019); Gur-Ari et al. (2018) show that the positive eigenvalues of the Hessian remain relatively stable throughout training, while the negative eigenvalues shrink rapidly. Alain et al. (2018) and Sagun et al. (2016) also show that the negative eigenvalues shift chaotically. Gur-Ari et al. (2018) show that when training a network on a classification task with k classes, then at least the eigenspace spanned by the k eigenvectors corresponding to the top k eigenvalues remains very stable. Sivan et al. (2024); Liu et al. (2024) also show that when training a neural network on a variety of tasks, the top k eigenvalues and their corresponding eigenvectors change very slowly. Alain et al. (2018) also show explicitly that the second-order Taylor approximation is a poor approximation of the loss function in the eigenspace corresponding to the negative eigenvalues (the concave eigenspace), but an excellent approximation in the eigenspace corresponding to the positive eigenvalues (the convex eigenspace); indeed, they show that the optimal step in the convex eigenspace is well estimated by the Newton step, while there is no correlation between the Hessian and the optimal step in the concave eigenspace. Using the Lipschitz parameter as a measure of the rate of change of the Hessian in a given subspace (hence a measure of the quality of a second-order Taylor approximation to a function and its corresponding Newton step), this supports the claim that $L^+ \ll L^-$.

K Lipschitz parameter experiments

K.1 Convex vs. concave Lipschitz parameters

We tested our algorithm in 7 scenarios with the PyTorch 1.13.0 framework, each on a single NVIDIA GeForce RTX 3090 GPU with the standard hyperparameters and settings for ARC:

- $\sigma_0^+ = \sigma_0^- = 1$
- $\eta_1 = 0.1, \eta_2 = 0.9$
- $\gamma_1 = \gamma_2 = 2$
- Maximum sub-problem failures = 11
- Maximum sub-problem iterations = 50,
- Sub-problem tolerance = 10^{-7}
- Lanczos eigendecomposition (Garber et al., 2016)
- BFGS trinomial sub-problem solver (Nocedal & Wright, 2006, Chapter 6.1), (Feinman, 2021)
- trinomial sub-problem maximal failures=11
- trinomial sub-problem maximal iterations=50

The test scenarios include combinations of:

- Training ResNet18 artificial neural networks (He et al., 2015) for image classification on MNIST, CIFAR10, and FakeData (random noise in place of images) with Cross-Entropy Loss (CELoss), to evaluate the effect of changing data on the Lipschitz parameters
- Training a CNN for image classification on CIFAR10 with CELoss, to evaluate the effect of changing neural network architecture on the Lipschitz parameters.
- Training a CNN[5] autoencoder[6] (LeCun, 1987) to compress MNIST

The classification CNN architecture:

1. A feature extractor consisting of 2 2D convolutional layers with 6 output channels for the first and 16 output channels for the second. Both had kernel sizes of 5 pixels. Each of these is followed by a ReLU nonlinearity and then 2x2 2D max pooling
2. A 3-layer MLP classification head with hidden sizes (120, 84) and ReLU nonlinearities

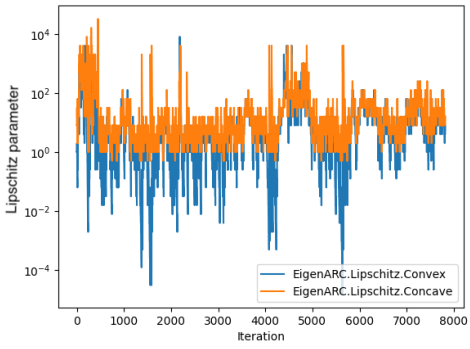
The autoencoder CNN architecture:

- Encoder: 4 2D convolutional layers with respective output channel numbers (16,32,32,64) and kernel sizes of 3 pixels for the first two and 5 pixels for the last two. The first two have 1 pixel padding and the last two have 2 pixel paddings. After every layer we apply LeakyReLU nonlinearity and after every 2 layers we apply 2x2 2D max pooling.
- Decoder: 4 composite layers consisting of
  1. a 2D transpose-convolutional layer
  2. LeakyReLU nonlinearity (only for first and third composite layers)
  3. a 2d convolutional layer
  4. LeakyReLU nonlinearity
- Decoder hidden channel sizes: 32-32-16-16-16-16-3
- Decoder kernel sizes: 2-5-5-5-2-3-5-3
- All strides are of size 1, except for the first and third transpose-convolutional layers of the decoder, with stride of size 2
- Decoder paddings: 0-2-2-2-0-1-2-1

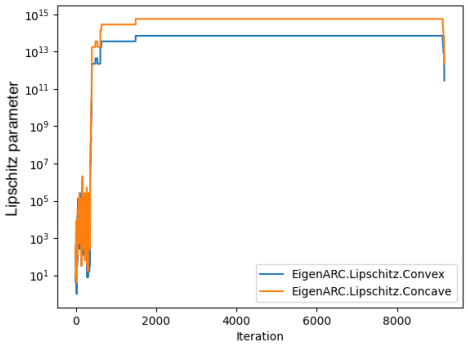
Each experiment took several hours to run. All experiments (including those from above) shown in figure 4.

[5] convolutional neural network

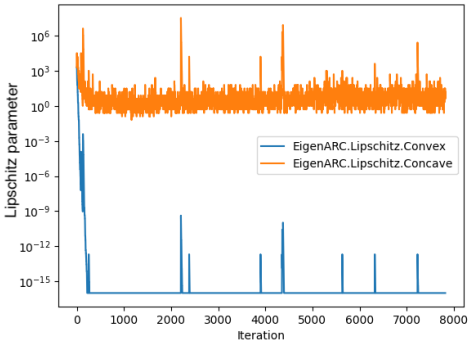
[6] With hidden dimensions 128-64-36-18-9-18-36-64-128, ReLU nonlinearities, and sigmoid nonlinearity on the output



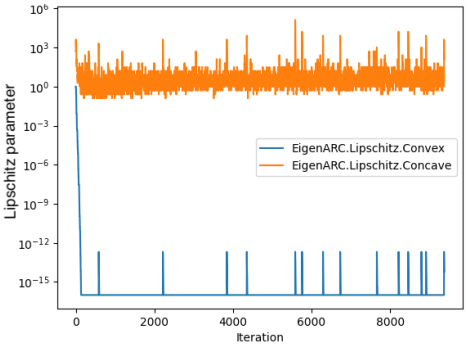
(a) Training autoencoder on CIFAR10



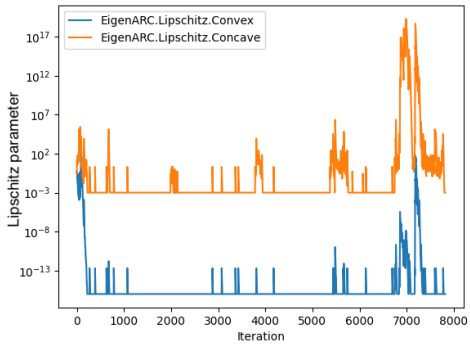
(b) Training autoencoder on MNIST



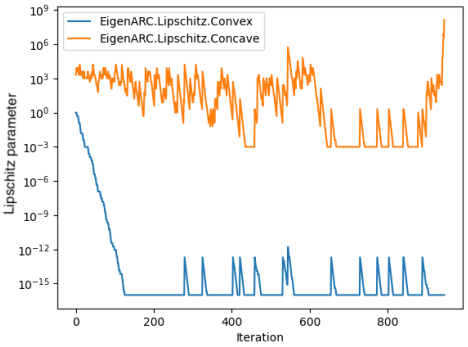
(c) Training Resnet on CIFAR10 with cross-entropy loss



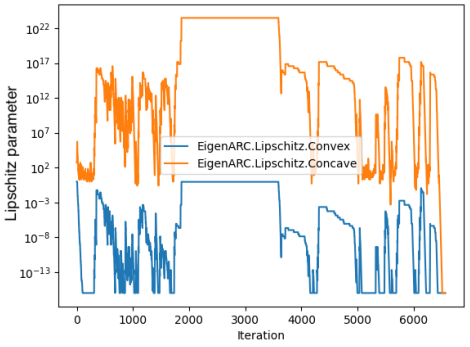
(d) Training Resnet on MNIST with cross-entropy loss



(e) Training CNN on CIFAR10 with cross-entropy loss



(f) Training Resnet on CIFAR10 with mean-squared error loss



(g) Training Resnet on random data (torchvision Fake-Data dataset) with cross-entropy loss

Figure 4: Comparisons of convex-subspace Lipschitz parameters to concave-subspace Lipschitz parameters. *Logarithmic scale*

One caveat is that due to computational constraints, we use stochastic minibatch training for the neural networks instead of using the full batch to compute the gradient and Hessian-vector products at each iteration (see Bertsekas (1996) for an introduction to minibatch Monte-Carlo estimation of a sum). However, Cartis et al. (2011a), notes that the adaptive Lipschitz parameter estimates may account for this variance by being greater than the actual Lipschitz parameters. Thus, our claims of $L^+ \ll 1$ are not affected (since our experiments effectively provide an upper bound on L^+) while our claims of $L^- \gg 0$ are weakened. Since there is no reason to expect the variance on L^- to be significantly greater than the variance on L^+, however, our experiments remain valid.

For visual clarity, the quasi-Newton superiority measurements in 3 are presented after:

1. Clipping extreme values to the 10% - 90% quantile range

2. Gaussian smoothing, consisting of a rolling window of size 300 and standard deviation of 100

K.1.1 Computation of Lipschitz parameters

We modified the standard ARC algorithm to compute distinct Lipschitz parameters for the eigenspaces corresponding to the minimal and maximal eigenvalues. Pseudocode for this algorithm is given below.

While lines 3-18 of `EigenARC` may technically be usable as the `LIPSCHITZ` subroutine of algorithm `ELMO` above, each iteration requires $\Omega(n)$ evaluations of the loss function, which will be computationally expensive if $n \gg 0$ and if the loss function is computationally heavy. This may be ameliorated by performing these calculations only for a small subset of the eigenspaces like Sivan et al. (2024), however we leave this to future work.

K.2 The effect of AMOS on top Lipschitz parameters

In Figure 5 we report the top-five Lipschitz parameters—corresponding to the most rapidly drifting curvature directions—throughout language-model training, as estimated by `AMOS`'s slope-EMA Lipschitz estimator. Ideally, `AMOS` would not alter these parameters, allowing its threshold to be adjusted independently of its effect on curvature variation. Nevertheless, we include these experiments for full disclosure and to avoid selective reporting. The estimated curvature variation is highest with the first-order optimizer (SGD) and lowest with the second-order optimizer (Sophia). Interestingly, well-tuned `AMOS` hybridization produces an dichotomic, split pattern between these two regimes.

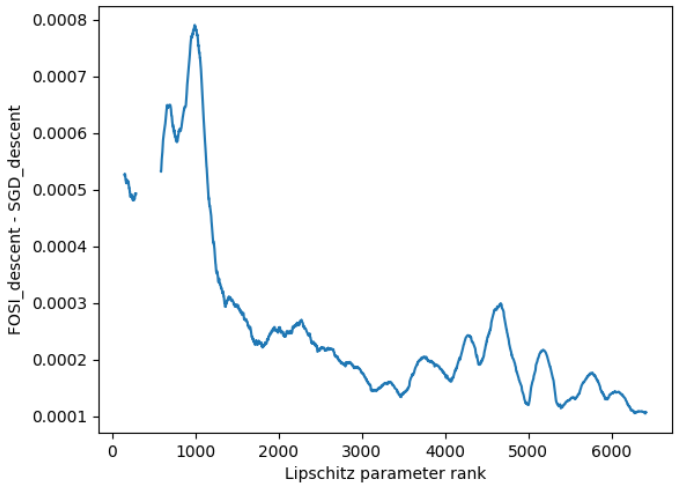


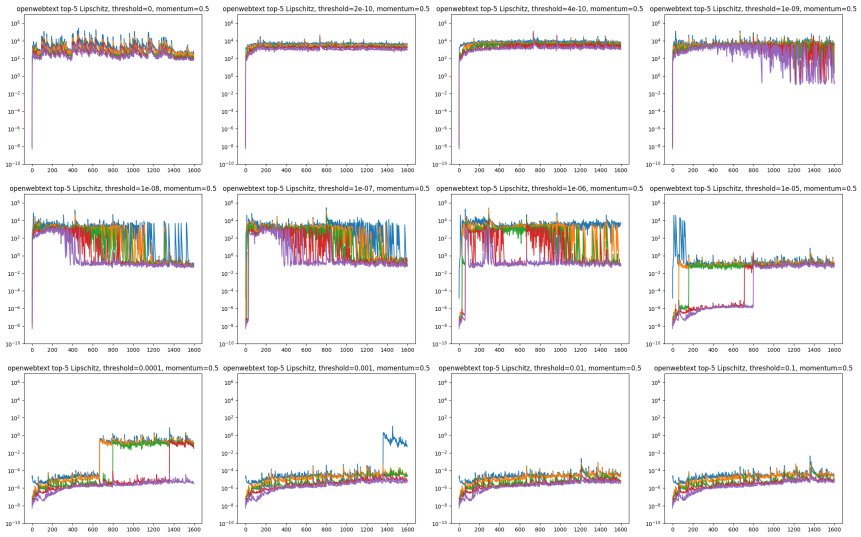
Figure 3: Inverse relation between a convex-subspace Lipschitz parameter and corresponding descent superiority of quasi-Newton method

Algorithm 3 Algorithm `EigenARC`

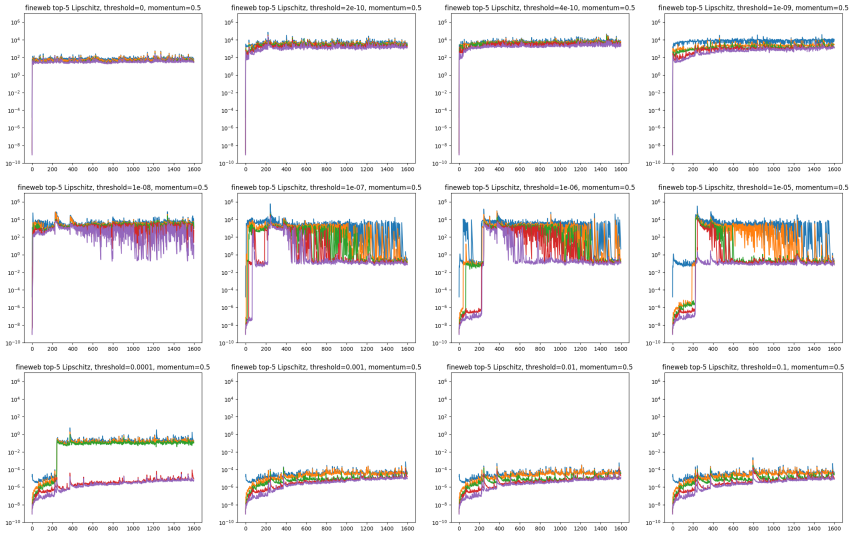
Require: $\epsilon \in \mathbb{R}^+, \theta_0 \in \mathbb{R}^n, \gamma_1 > 1 > \gamma_2 > 0, \eta_2 \geq \eta_1 > 0, \left(L_0^i\right)_{i=1}^n > 0$, `EIGEN`, `BASE_OPT`

$t \leftarrow 0$
while $\|\nabla f(\theta_t)\|_2 > \epsilon$ **do**
  {While `BASE_OPT` hasn't converged yet}
  $(\lambda_i, v_i)_{i=1}^n \leftarrow$ `EIGEN` $(\mathcal{H}(\theta_t))$
  if `ASSESS_LIPSCHITZ` $\left((L_t^i)_{i=1}^n\right) > \eta_2$ **then**
    {Overly conservative L_t^i}
    $\left(L_t^i\right)_{i=1}^n \leftarrow \gamma_2 \cdot \left(L_t^i\right)_{i=1}^n$
  else
    if `ASSESS_LIPSCHITZ` $\left((L_t^i)_{i=1}^n\right) < \eta_1$ **then**
      {Overly liberal L_t^i}
      while `ASSESS_LIPSCHITZ` $\left((L_t^i)_{i=1}^n\right) > \eta_1$ **do**
        {Raise all L_t^i assessment is passed}
        $\left(L_t^i\right)_{i=1}^n \leftarrow \gamma_1 \cdot \left(L_t^i\right)_{i=1}^n$
      end while
      for i=1,...,n **do**
        {Reduce the L_t^i that can be reduced without violating assessment}
        while `ASSESS_LIPSCHITZ` $\left((L_t^i)_{i=1}^n\right) > \eta_1$ **do**
          $L_t^i \leftarrow \frac{L_t^i}{\gamma_1}$
        end while
        $L_t^i \leftarrow \gamma_1 \cdot L_t^i$
      end for
    end if
  end if
  $\theta_{t+1} \leftarrow$ `BASE_OPT` (θ_t)
  $t \leftarrow t + 1$
end while
return $\left(L_{\hat{t}}^i\right)_{i=1,\hat{t}=1}^{n,t}$

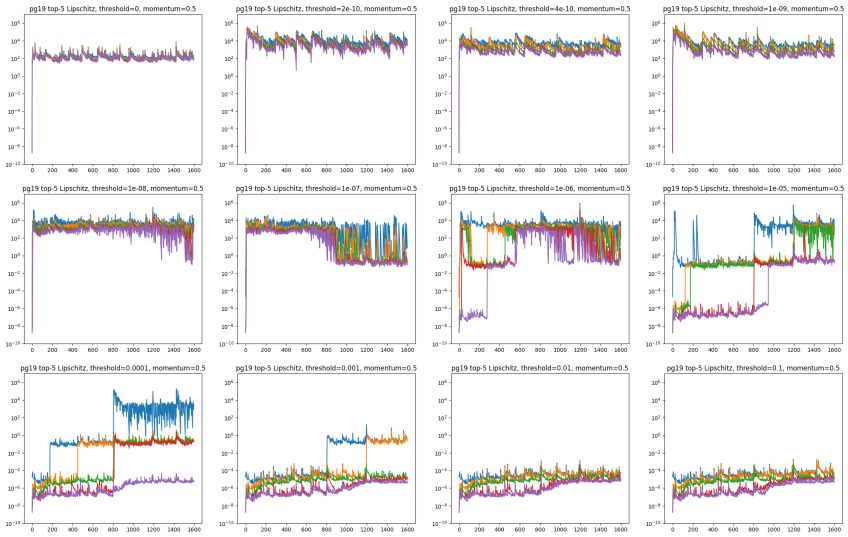
Function ASSESS_LIPSCHITZ$\left(\hat{L_t^i}\right)_{i=1}^n$:
return $\frac{f(\theta_t) - f\left(\theta_t + \sum_{i=1}^n \Delta\theta_t^{*\top} v_i \left(\hat{L_t^i}\right) \cdot v_i\right)}{-\sum_{i=1}^n M_t^i \left(\Delta\theta_t^{*\top} v_i \left(\hat{L_t^i}\right)\right)}$



(a) Training GPT-2 on OpenWebText across different AMOS Lipschitz thresholds



(b) Training GPT-2 on FineWeb-Edu across different AMOS Lipschitz thresholds



(c) Training GPT-2 on PG19 across different AMOS Lipschitz thresholds

Figure 5: **Top-five Lipschitz parameters during training.** Estimated curvature variation for SGD, Sophia, and AMOS. AMOS exhibits an intermediate split pattern between the first- and second-order baselines. The x-axes show the number of Lipschitz-parameter recomputations, and the y-axes show the corresponding Lipschitz-parameter magnitudes.

L EXPERIMENT DETAILS

We apply AMOS in practice to train our GPT-2 model on the FineWeb-Edu dataset, TinyShakespeare dataset, OpenWebText dataset, and PG19 dataset on an Nvidia RTX 2080 Ti GPU.

- We trained 6-layer transformer with 8-head multihead attention on the TinyShakespeare dataset (Karpathy, 2015) for 2,010 training iterations, and on the Fineweb-Edu (Penedo et al., 2024) dataset, OpenWebText dataset (Gokaslan et al., 2019), and PG19 dataset (Rae et al., 2019) for 16,010 iterations.
- Our first 6 baselines (AdamW, SOAP, LBFGS, Muon, SGD, Sophia) take the tuned hyperparameters from vanilla NanoGPT, namely - learning rate = 10^{-4}, weight decay = 10^{-2}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, with weight decay on tensors of dimension 2 or higher (weight decay and betas applied where relevant). We experimented a little with random search to find better hyperparameters for our baselines, but didn't notice any significant difference in the results, so to maintain a fair comparison - we did not tune AMOS's hyperparameters on any of the latter 3 datasets, instead relying solely on settings tuned on the Tiny Shakespeare dataset (details on this a few items down); in this sense we used AMOS's "default" hyperparameters as well. For Muon we followed recommended practice of applying a larger learning rate than AdamW, namely 0.02. MADA, FOSI, and AdaCubic take their hyperparameters from the default settings in their respective repos.
- Batch size = 4. Batch size 4 was the largest batch that allowed the model, activations, optimizer states, and curvature computations to fit on the 11 GB RTX 2080 Ti under the common experimental configuration.
- Context size = 1024
- Embedding dimension = 512
- Random seed = 1337
- FP16 precision

- Hessian computation every 10 iterations (where relevant)

- Validation loss estimated every 200 iterations

- AMOS's first-order optimizer is SGD with learning rate set to $1.25 \cdot 10^{-}5$, selected by grid hyperparameter search. We also tried adding a Cosine Annealing scheduler for SGD, but this harmed SGD's performance and had no qualitative effect on the training trajectory comparison graph, so it is not included in our results.

- AMOS's second-order optimizer is Sophia, with $\rho = 0.01$ and otherwise Karpathy (2022)'s standard AdamW hyperparameters.

- Our Sophia and SGD reference optimization trajectories use optimizers with the same hyperparameters but without dynamic optimizer selection (instead using pure SGD all the way through or pure Sophia).

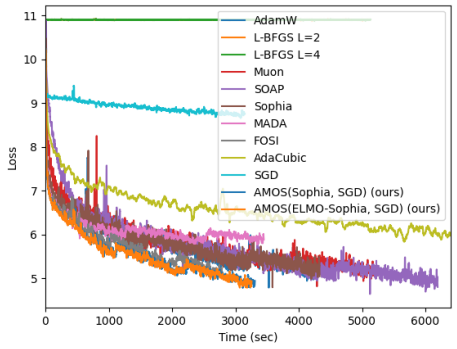
- For all datasets we set the momentum coefficient for our meta-optimizer $\alpha_L = 0.5$, and the Lipschitz threshold $L_{thresh} = 10^{-7}$, $\epsilon_2 = 10^{-12}$, determined by grid search on the TinyShakespeare dataset alone.

- For hardware efficiency reasons, we transfer parameters to SGD at a per-tensor granularity instead of per-scalar. The mean Lipschitz parameter of the entire tensor is compared against the preset threshold to determine if it is transferred to the first-order optimizer or not.

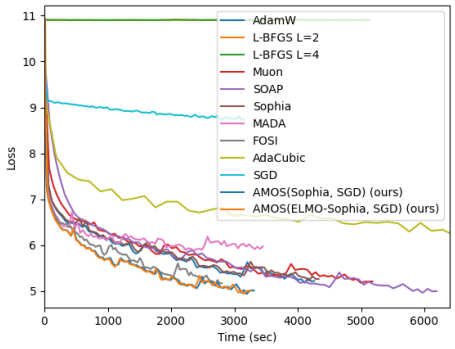
- For visual clarity, the training graphs in 6 are presented after Savitzky-Golay linear smoothness filtering with window length = 50 (or =500 for PG19 training loss, due to significantly more variance in the graph).

- For the ELMO-based Hessian eigenvalue regularization rule, we computed the regularized eigenvalues using the gradient available at the time of Hessian evaluation, rather than re-regularizing them at every iteration with the current gradient. The resulting regularized Hessian was then held fixed until the next Hessian update to reduce per-step computational complexity.

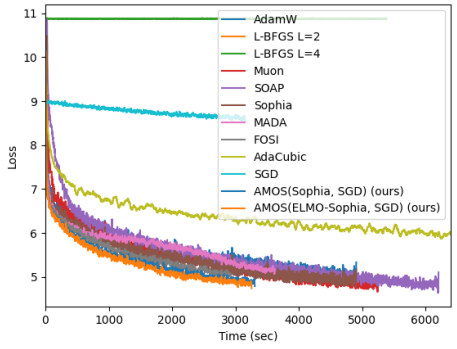
Experiment results are presented in Figure 6. We can see that AMOS consistently outperforms the alternative optimizers, both in wall-clock time and in iteration count. The only optimizer that comes close is FOSI on the PG19 dataset, which was trained on a much more powerful GPU (A6000) due to OOM on the RTX 2080 Ti that the remaining baselines were run on. However we note that FOSI consistently outperforms MADA while trained on the same hardware, which may be partially explained by appendix C, where we discuss the regret near-optimality of FOSI.



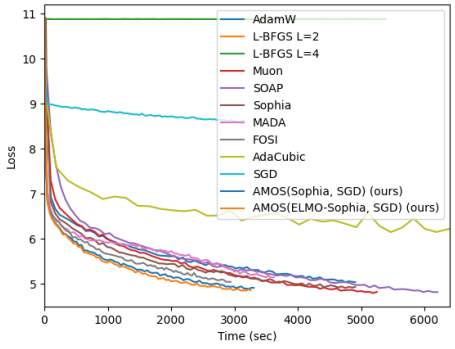
(a) Train loss: FineWeb-Edu dataset



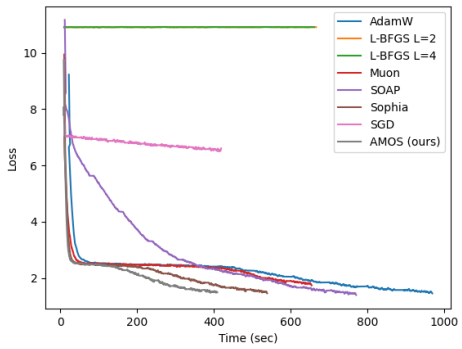
(b) Test loss: FineWeb-Edu dataset



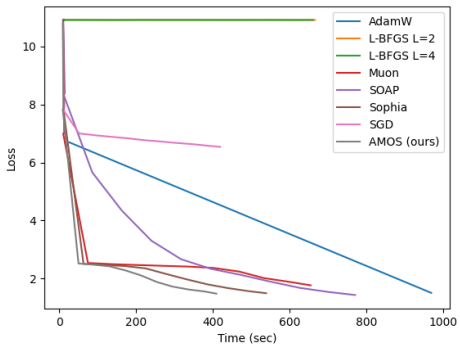
(c) Train loss: OpenWebText dataset



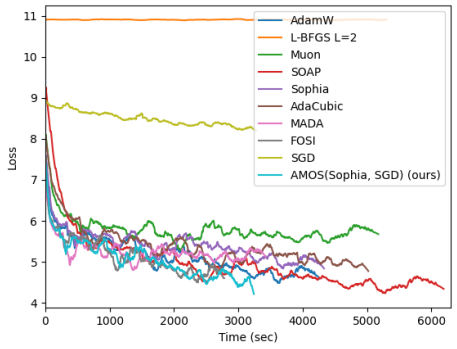
(d) Test loss: OpenWebText dataset



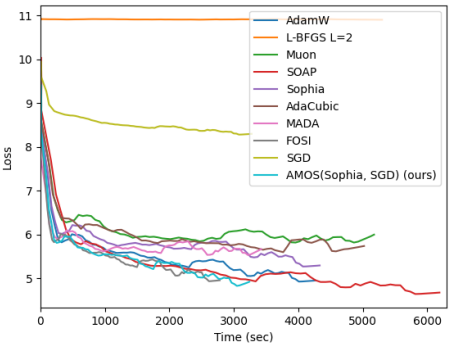
(e) Train loss: TinyShakespeare dataset



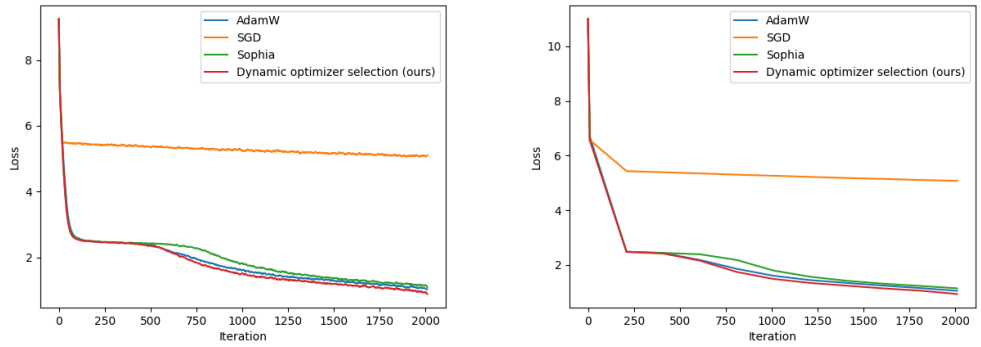
(f) Test loss: TinyShakespeare dataset



(g) Train loss: PG19 dataset



(h) Test loss: PG19 dataset

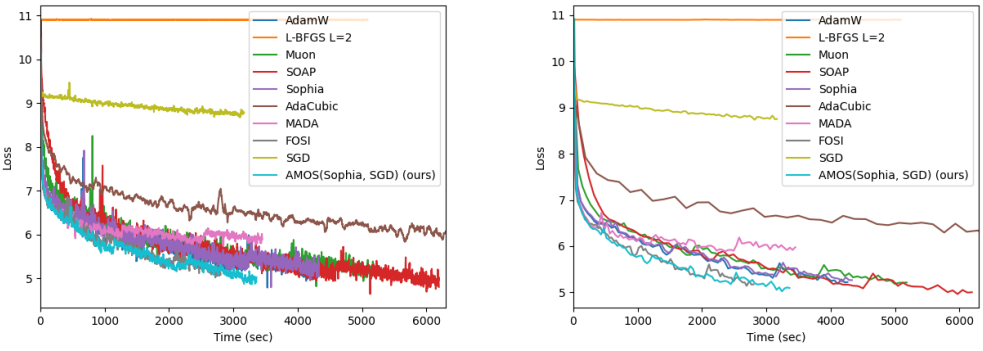


(i) Train loss: TinyShakespeare dataset (plotted against iteration count)

(j) Test loss: TinyShakespeare dataset (plotted against iteration count)

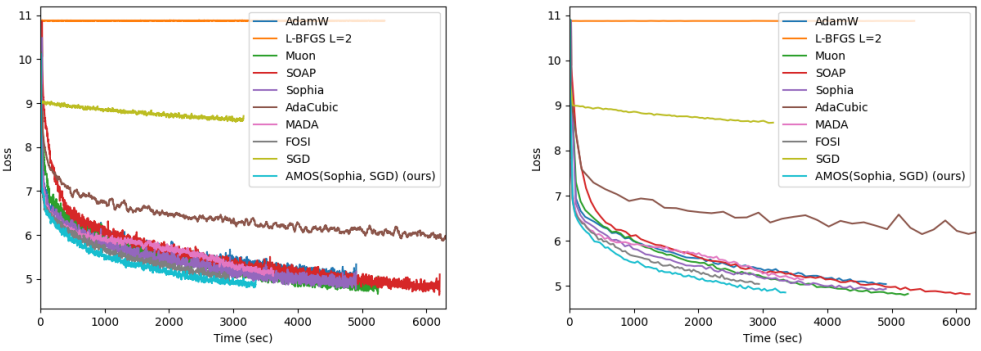
Figure 6: Comparison of AMOS-combined SGD and Sophia vs. pure optimizers. AMOS consistently outperforms the pure alternatives.

We present experiments with a different random seed (1336) to demonstrate robustness to randomness in figure 7.



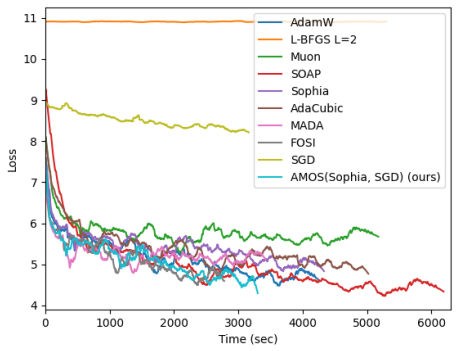
(a) Train loss: FineWeb-Edu dataset

(b) Test loss: FineWeb-Edu dataset

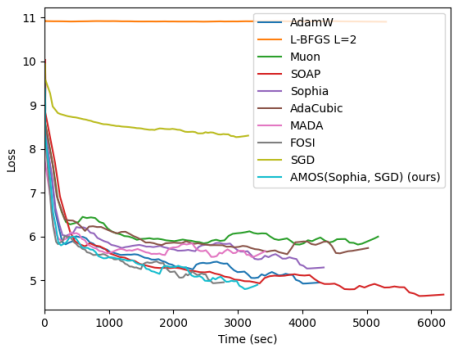


(c) Train loss: OpenWebText dataset

(d) Test loss: OpenWebText dataset



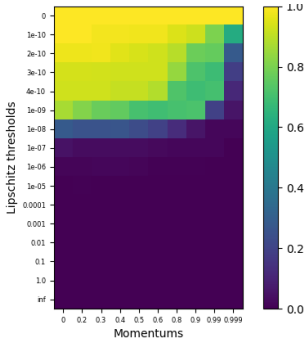
(e) Train loss: PG19 dataset



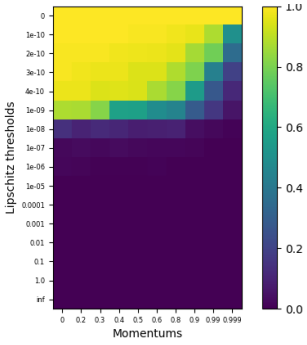
(f) Test loss: PG19 dataset

Figure 7: Comparison of AMOS-combined SGD and Sophia vs. pure optimizers on a different seed. AMOS consistently outperforms the pure alternatives.

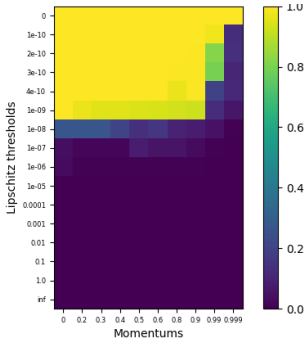
We explore the effect of AMOS hyperparameter selection on average second-order optimizer sparsity (i.e. on what percentage of the model is assigned to the second-order optimizer) in Figure 8. We see that the selected hyperparameters (chosen with grid search on the TinyShakespeare dataset alone) opt for a balanced sparsity, assigning some parameters to the second-order optimizer for improved convergence properties while assigning others to the first-order optimizer for reduced iteration cost.



(a) FineWeb-Edu dataset sparsity



(b) OpenWebText dataset sparsity



(c) TinyShakespeare dataset sparsity

Figure 8: Second-order optimizer mean sparsity levels for different hyperparameter settings. A higher value means fewer parameters assigned to the second-order optimizer.

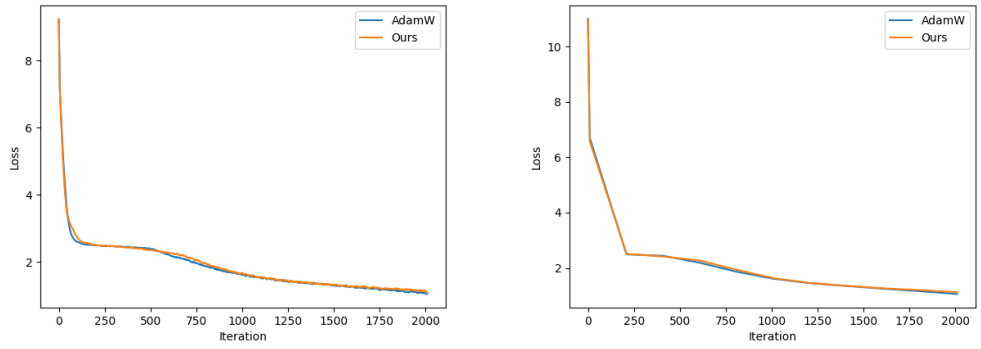
Most momentums do not significantly affect sparsity, however we note that our Lipschitz initialization scheme (initialization to 0) biases the Lipschitz parameters towards 0, with greater momentums conferring greater bias. The result is that the highest momentums reduce sparsity by biasing the Lipschitz parameters to being under the threshold, particularly at the beginning of the training process

(this is to be compared to RMSProp's (Tieleman & Hinton, 2012) bias towards 0, later fixed by Adam (Kingma & Ba, 2014)). We leave improved initialization schemes to future work.

M HYPERPARAMETER SENSITIVITY EXPERIMENTS

To demonstrate AMOS's robustness to hyperparameter selection, we hold Lipschitz threshold at 10^{-7} and present experiments demonstrating the effect of different momentum settings in Figure 10. We see that momentum has little effect on training trajectory, although the highest momentums do somewhat slow convergence (in wall-clock time) at first by reducing second-order optimizer sparsity, as noted in Appendix L.

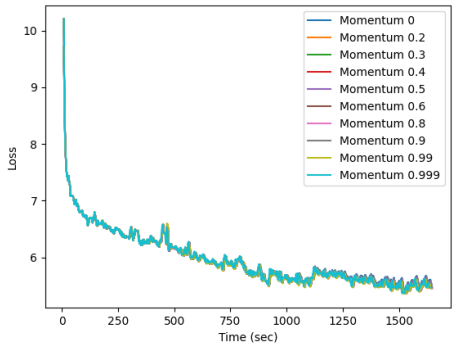
Due to the effect of AMOS's Lipschitz parameter threshold on second-order optimizer sparsity, AMOS is somewhat sensitive to threshold choice (although it is not overly sensitive - setting the threshold to values around 10^{-7} gave us good results on all datasets). Although setting the threshold to 10^{-7} gave us good results in all our experiments, we offer a second method for contending with this sensitivity - setting the threshold L_{thresh} to equation 8 (thus requiring no tuning). We rerun the TinyShakespeare tests with this setting, and compare the training- and test-loss of the experiment against the aforementioned standard AdamW baseline in Figure 9. It is evident that the second-order optimizer sparsity conferred by AMOS does not harm the convergence rate.



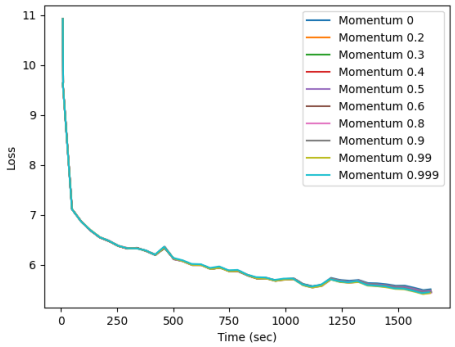
(a) Train loss: dynamic threshold selection (plotted against iteration count)

(b) Test loss: dynamic threshold selection (plotted against iteration count)

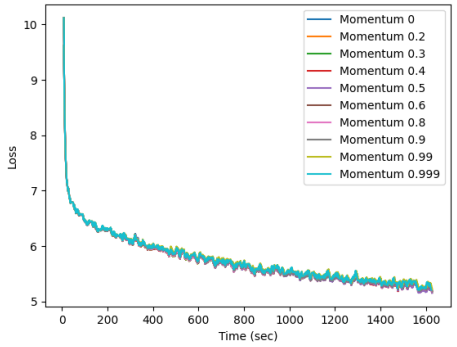
Figure 9: Comparison of AMOS-combined SGD and Sophia vs. AdamW. We see little difference between the two despite offloading 16% of parameters to cheaper SGD optimizer.



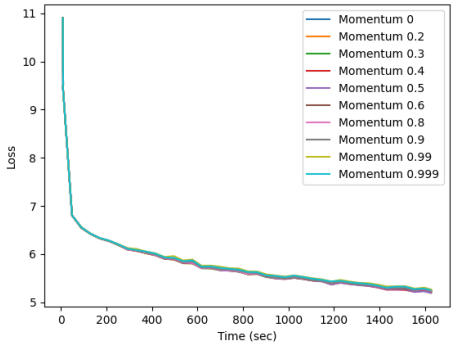
(a) Train loss: FineWeb-Edu dataset



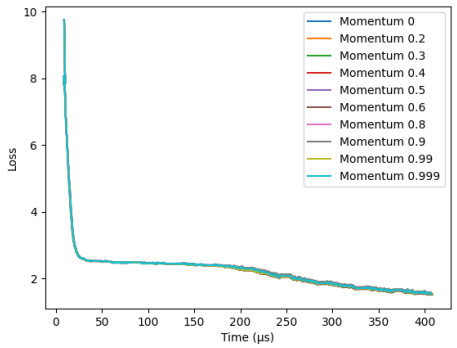
(b) Test loss: FineWeb-Edu dataset



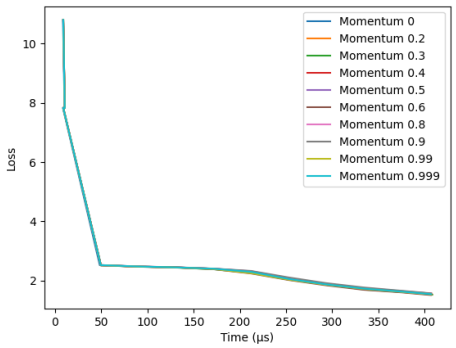
(c) Train loss: OpenWebText dataset



(d) Test loss: OpenWebText dataset



(e) Train loss: TinyShakespeare dataset



(f) Test loss: TinyShakespeare dataset

Figure 10: AMOS's sensitivity to momentum selection. Lipschitz threshold held at 10^{-7}.

The settings for this experiment are identical to those above, with the following exceptions:

- SGD's learning rate reduced to $3.125 \cdot 10^{-7}$
- $\epsilon_2 = 10^{-15}$
- Momentum $\alpha_L = 0.7$
- Each tensor's Lipschitz parameter is chosen with maximum aggregation across the tensor instead of mean aggregation

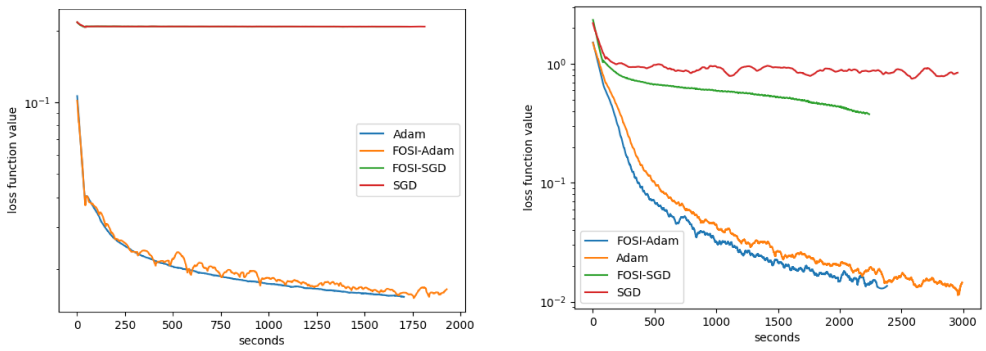
N LIPSCHITZ-AIDED OPTIMIZER SELECTION

In this section, we demonstrate the use of convex Lipschitz parameters to select the best optimizer for our use case. Due to the relative constancy of Lipschitz parameters throughout the training process (after an initial warmup phase) in different settings, we can select optimizers for each

setting based on the following rule: **quasi-Newton optimizers hold an advantage over first-order optimizers when the convex Lipschitz parameters are small**. As discussed in section 6, the convex Lipschitz parameters in the image autoencoder training setting are far larger than those in the image classification setting, so we compare a quasi-Newton optimizer against first-order methods in these settings to validate our rule.

FOSI Sivan et al. (2024) is a variant of Saddle-Free Newton Dauphin et al. (2014) which applies Newton iterations in the domain space subspaces spanned by the dominant eigenvectors of the Hessian, and a first-order "base optimizer" in the remaining subspaces. We use FOSI as our representative second-order optimizer due to its computational effectiveness, capability to adjust the computational complexity of each iteration by adjusting the number of "dominant" eigenvectors to compute (fewer eigenvectors comes at the cost of a poorer Hessian approximation by approximating the Hessian with a lower-rank matrix, although this is somewhat mitigated by applying the base optimizer in these subspaces), and fairness of comparison (since its integration of first-order optimizers allows us to compare the effect of second-order optimization in the dominant eigenspaces against first-order optimization in these spaces, while all else is held equal - the remaining subspaces are both treated by the same first-order optimizers).

Experiment results may be seen in figure 11.



(a) The setting in which we train a CNN on an autoencoder task has large convex Lipschitz parameters throughout training

(b) The setting in which we train ResNet18 on a classification task has small convex Lipschitz parameters throughout training

Figure 11: Comparison of second-order optimizers against first-order optimizers in settings with different sized convex Lipschitz parameters. Second-order optimizers only hold an advantage over first-order optimizers (thus justifying their additional computational complexity) when the convex Lipschitz parameters are small.

The experiments are run with the same settings as before, with FOSI augmenting SGD and Adam respectively and Savitzky-Golay order-2 filtering with a window size of 5000 for clarity of visualization. It may be clearly seen that FOSI second-order augmentation is beneficial only in the classification setting, due to the small convex Lipschitz parameters.

O PAC JUSTIFICATION FOR AMOS'S SINGLE-SLOPE ESTIMATE

In AMOS, the per-eigenspace Lipschitz parameter is the maximum possible eigenvalue slope. We view a slope measurement as a random variable Z, and use one observed slope $\widehat{Z} \sim Z$ to estimate this maximum. For example, AMOS obtains $\widehat{Z}$ from the finite difference of an eigenvalue over two successive iterates (with the stabilizer ϵ_2 in the denominator). The following result justifies this as an estimate of a functionally near-maximal slope under a low-variance assumption.

Let

$$\nu := \mathbb{E}[Z], \qquad \mathrm{Var}(Z) = \tau^2, \qquad \mathcal{L} := \operatorname{ess\,sup} Z.$$

Here, $\mathcal{L}$ is the Lipschitz parameter that AMOS seeks to estimate. Finite variance alone cannot bound $\mathcal{L} - \nu$, since an arbitrarily large slope can occur with arbitrarily small probability. We therefore use a high quantile as a functionally near-maximal Lipschitz value.

For $q \in (0, 1)$, let

$$u_q := \inf\{u : \Pr(Z \le u) \ge q\}$$

be the q-quantile of the slope distribution. At most a $1 - q$ fraction of slopes are larger than u_q; hence, for high q, u_q is a functionally near-maximal slope.

By Cantelli's inequality,

$$\Pr(Z - \nu \ge a) \le \frac{\tau^2}{\tau^2 + a^2}.$$

Setting $a = \tau\sqrt{\frac{q}{1-q}}$ gives

$$u_q \le \nu + \tau\sqrt{\frac{q}{1-q}} \tag{17}$$

Applying Cantelli's inequality to the lower tail of the single observed slope $\widehat{Z}$ gives, for every $\delta \in (0, 1)$,

$$\Pr\left[\widehat{Z} \ge \nu - \tau\sqrt{\frac{1-\delta}{\delta}}\right] \ge 1 - \delta \tag{18}$$

Combining equation 17 and equation 18 yields

$$\boxed{\Pr\left[u_q - \widehat{Z} \le \tau\left(\sqrt{\frac{q}{1-q}} + \sqrt{\frac{1-\delta}{\delta}}\right)\right] \ge 1 - \delta.}$$

Equivalently, the one-slope AMOS estimate is an (ε, δ)-PAC lower estimator of the functionally near-maximal Lipschitz value u_q whenever

$$\tau\left(\sqrt{\frac{q}{1-q}} + \sqrt{\frac{1-\delta}{\delta}}\right) \le \varepsilon.$$

Thus, when slope measurements have low variance, a single slope is, with high probability, close to a value exceeded by only a small fraction of slopes.

P PROOFS

P.1 PRELIMINARY LEMMAS

Before we can get started, we prove a few basic lemmas.

Lemma P.1.

$$\forall_{x \ge -1} : \sqrt{1+x} \le 1 + \frac{x}{2}$$

Proof. Mark $g : \mathbb{R} \to \mathbb{R}, g(x) = 1 + \frac{x}{2} - \sqrt{1+x}$. We have

$g'(x) = \frac{1}{2}\left(1 - \frac{1}{\sqrt{1+x}}\right)$

$g''(x) = \frac{1}{4}\left(\frac{1}{(1+x)^{\frac{3}{2}}}\right)$

g is convex due to its second derivative being positive for all $x > -1$. Therefore, its sole critical point $x = 0$ obtained from the derivative is a minimum, and $\forall_{x \geq -1} : g(x) \geq g(0) = 0$ □

Corollary P.1.1.

$$\forall_{x \in \mathbb{R}^+} \forall_{y \geq -x} : \sqrt{x+y} \leq \sqrt{x} + \frac{y}{2\sqrt{x}}$$

Proof.

$$\sqrt{x+y} = \sqrt{x}\sqrt{1+\frac{y}{x}} \leq \sqrt{x}\left(1 + \frac{y}{2x}\right) = \sqrt{x} + \frac{y}{2\sqrt{x}}$$

□

Lemma. *Let $f : \mathbb{R}^n \to \mathbb{R}$ satisfy assumptions 1 and 2. Then*

$$m_t^i\left(\Delta\theta_t^{*\top} v_i\right) \leq M_t^i\left(\Delta\theta_t^{*\top} v_i\right) \leq 0 \tag{19}$$

Proof. The first inequality stems from the trivial fact that $m_t^i \leq M_t^i$.

The second inequality follows from the fact that (by design), $\Delta\theta_t^{*\top} v_i$ is a minimizer of $M_t^i\left(\Delta\theta_t^{\top} v_i\right)$, but

$$M_t^i(0) = 0$$

□

Corollary P.1.2. *Each ELMO step monotonically decreases the loss function value.*

Proof. Immediate from combining equation 19 with Lemma 3.1. □

Lemma. *3.2 Minmax preconditioner*

$$\underset{\Phi_t \in \mathbb{R}^{n \times n}}{\arg\min} \left|\Delta\Delta^i\theta_t(\Phi_t)\right| = \left(\frac{\mathcal{H}(\theta_t) + \sqrt{(\mathcal{H}(\theta_t))^2 + 2V \cdot D \cdot V^\top}}{2}\right)^{-1}$$

with

$$D = diag\left(L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|\right)_{i=1}^n \succeq 0$$

Proof.

$$\begin{aligned}\left|\Delta\Delta^i\theta_t\right| &= \left|\nabla f(\theta_t)^\top \left(\Phi_t - \frac{2}{\lambda_i + \sqrt{\lambda_i^2 - 2L_t^i \cdot \nabla f(\theta_t)^\top v_i}} I\right) v_i\right| \\ &= \left|\nabla f(\theta_t)^\top \left(\Phi_t - \left(\frac{\mathcal{H}(\theta_t) + \sqrt{(\mathcal{H}(\theta_t))^2 + 2D_i I}}{2}\right)^{-1}\right) v_i\right|\end{aligned}$$

with

$$D_i = L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|$$

and the result follows from developing the second parenthesized term for all n dimensions of the domain space. □

P.2 LEMMA 3.1: EIGENSPACE DESCENT

Working with assumptions 1 and equation 2, Dennis & Schnabel (1983, Lemma 4.1.14) prove the following:

Lemma P.2.

$$f(\theta_{t+1}) - f(\theta_t) \leq \nabla f(\theta_t)^T \cdot \Delta\theta_t + \frac{1}{2}\Delta\theta_t^T \mathcal{H}(\theta_t)\Delta\theta_t + \frac{1}{6}L_H \|\Delta\theta_t\|_2^3 \tag{20}$$

Much like algorithm ELMO, minimizing equation 20 would maximize [a bound on] the descent given by iteration t. However, previous works such as Nesterov & Polyak (2006) note the difficulty of minimizing this 3rd-order n-dimensional polynomial, even when L_H is known. Indeed, Cartis et al. (2011a) propose minimizing it iteratively over a growing subspace, with each iteration's minimization subspace a superset of the previous iterations' (in practice, they use the Hessian's Krylov subspaces, initialized with the gradient). In our theoretical analysis however, we have the freedom to simply take the most natural decomposition of the space into subspaces, the eigenspaces of the Hessian. This does not limit the practicality of our approach, however, since Lanczos methods allow one to obtain elements of this decomposition. In fact, Sivan et al. (2024) demonstrate experimentally that decomposing the parameter space into multiple eigenspaces and optimizing each separately can significantly speed up optimization wall time, despite the additional computational burden of the Lanczos iterations, because of the regularizing effect this has on the function in each of the subspaces (by reducing the variance of the Hessian eigenvalues). Ghorbani et al. (2019) also show the benefits of reducing this variance.

Lemma. *3.1 Eigenspace Descent Bounds*

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function satisfying assumptions 1 and 2, and let $\theta_{t+1} \in \mathbb{R}^n$. Marking $\Delta\theta_t = \theta_{t+1} - \theta_t$, we have

$$\exists_{\left(L_t^i\right)_{i=1}^n \in (\mathbb{R}^+)^n} : f(\theta_{t+1}) - f(\theta_t) \leq \sum_{i=1}^n M_t^i \left(\Delta\theta_t^\top v_i\right) \tag{21}$$

$$\exists_{\left(L_t^i\right)_{i=1}^n \in (\mathbb{R}^+)^n} : f(\theta_{t+1}) - f(\theta_t) \geq \sum_{i=1}^n m_t^i \left(\Delta\theta_t^\top v_i\right) \tag{22}$$

We give 2 proofs of the above lemma. The first proof is far simpler and relies on the standard spectral norm-Lipschitz continuous Hessian assumption given by equation 2 instead of assumption 2:

Proof. Beginning with Nesterov & Polyak (2006, Lemma 1) for the first inequality,

$$
\begin{aligned}
&\left|f\left(\theta_{t+1}\right)-f\left(\theta_t\right)-\left(\nabla f\left(\theta_t\right)^{\top}\left(\theta_{t+1}-\theta_t\right)+\left(\theta_{t+1}-\theta_t\right)^{\top}\mathcal{H}\left(\theta_t\right)\left(\theta_{t+1}-\theta_t\right)\right)\right| \\
&\leq L_H\left\|\theta_{t+1}-\theta_t\right\|_2^3 \\
&\leq\left|f\left(\theta_{t+1}\right)-f\left(\theta_t\right)-\sum_{i=1}^n\left(\nabla f\left(\theta_t\right)^{\top} v_i \cdot\left(\theta_{t+1}-\theta_t\right)^{\top} v_i+\lambda_i\left(\left(\theta_{t+1}-\theta_t\right)^{\top} v_i\right)^2\right)\right| \\
&\leq L_H\left\|\sum_{i=1}^n\left(\theta_{t+1}-\theta_t\right)^{\top} v_i v_i^{\top}\right\|_2^3 \\
&\leq L_H\left(\sum_{i=1}^n\left\|\left(\theta_{t+1}-\theta_t\right)^{\top} v_i v_i^{\top}\right\|_2\right)^3 \\
&=L_H \cdot n^3\left(\sum_{i=1}^n \frac{1}{n}\left|\left(\theta_{t+1}-\theta_t\right)^{\top} v_i\right|\right)^3 \\
&\leq L_H \cdot n^3\left(\frac{1}{n} \sum_{i=1}^n\left|\left(\theta_{t+1}-\theta_t\right)^{\top} v_i\right|^3\right) \\
&=L_H \cdot n^2 \sum_{i=1}^n\left|\left(\theta_{t+1}-\theta_t\right)^{\top} v_i\right|^3 \\
&=\sum_{i=1}^n \widetilde{L_H} \cdot\left|\left(\theta_{t+1}-\theta_t\right)^{\top} v_i\right|^3
\end{aligned}
$$

with

1. the second inequality being a representation of $\theta_{t+1}-\theta_t$ over the (orthogonal) Hessian eigenbasis
2. the third inequality due to the triangle inequality
3. the fourth inequality due to Jensen's inequality
4. $\widetilde{L_H} \triangleq L_H \cdot n^2$

□

Our first proof of lemma 3.1 is simple, but leaves something to be desired due to its lack of per-eigenspace Lipschitz parameters and due to the presence of n^2 in the bound, which can be very large, as noted in section A. The first proof's assumption of equation 2 is also easily seen to be no weaker than assumption 2 (meaning that assuming equation 2 implies assumption 2) by taking $L_R \triangleq L_t^i \triangleq L_H$ for all $t \in [T], i \in [n]$. To address these concerns, we make use of assumption 2, and give a second (though more complicated) proof for lemma 3.1. We'll prove only the upper bound, as a proof for the lower bound is similar.

Proof.

$$
\begin{aligned}
&f(\theta_{t+1}) - f(\theta_t) \\
&= \int_0^1 \nabla f(\theta_t + y(\theta_{t+1} - \theta_t))^\top (\theta_{t+1} - \theta_t)\, dy \\
&= \nabla f(\theta_t)^\top (\theta_{t+1} - \theta_t) + \int_0^1 (\nabla f(\theta_t + y(\theta_{t+1} - \theta_t)) - \nabla f(\theta_t))^\top (\theta_{t+1} - \theta_t)\, dy \\
&= \nabla f(\theta_t)^\top (\theta_{t+1} - \theta_t) + \int_0^1 \left(\int_0^1 y \mathcal{H}(\theta_t + yz(\theta_{t+1} - \theta_t))(\theta_{t+1} - \theta_t)\, dz \right)^\top (\theta_{t+1} - \theta_t)\, dy \\
&= \nabla f(\theta_t)^\top (\theta_{t+1} - \theta_t) + \int_0^1 \int_0^1 y(\theta_{t+1} - \theta_t)^\top \mathcal{H}(\theta_t + yz(\theta_{t+1} - \theta_t))(\theta_{t+1} - \theta_t)\, dydz \\
&= \nabla f(\theta_t)^\top (\theta_{t+1} - \theta_t) + (\theta_{t+1} - \theta_t)^\top \mathcal{H}(\theta_t)(\theta_{t+1} - \theta_t) \\
&+ \underbrace{\int_0^1 \int_0^1 y(\theta_{t+1} - \theta_t)^\top (\mathcal{H}(\theta_t + yz(\theta_{t+1} - \theta_t)) - \mathcal{H}(\theta_t))(\theta_{t+1} - \theta_t)\, dydz}_{Z}
\end{aligned}
$$

with the first and third equalities due to the fundamental theorem of calculus.

Mark the Hessian eigendecompositions as follows:

$$
\begin{aligned}
\mathcal{H}(\theta_t + yz(\theta_{t+1} - \theta_t)) &= \tilde{V}\tilde{\Lambda}\tilde{V}^\top \\
\mathcal{H}(\theta_t) &= V\Lambda V^\top
\end{aligned}
$$

with diagonal $\Lambda = diag(\lambda_i)_{i=1}^n, \tilde{\Lambda} = diag\left(\tilde{\lambda}_i\right)_{i=1}^n$ and orthogonal (due to the Hermitian nature of Hessian matrices) matrices $V, \tilde{V}$.

$$
\begin{aligned}
Z &= \int_0^1 \int_0^1 y(\theta_{t+1} - \theta_t)^\top \left(\tilde{V}\tilde{\Lambda}\tilde{V}^\top - V\Lambda V^\top\right)(\theta_{t+1} - \theta_t)\, dydz \\
&= \int_0^1 \int_0^1 y(\theta_{t+1} - \theta_t)^\top \left(V\tilde{\Lambda}V^\top - V\Lambda V^\top\right)(\theta_{t+1} - \theta_t)\, dydz \\
&+ \int_0^1 \int_0^1 y(\theta_{t+1} - \theta_t)^\top \left(\tilde{V}\tilde{\Lambda}\tilde{V}^\top - V\tilde{\Lambda}V^\top\right)(\theta_{t+1} - \theta_t)\, dydz
\end{aligned}
$$

Focusing on the first term,

$$
\begin{aligned}
&= \int_0^1 \int_0^1 y(\theta_{t+1} - \theta_t)^\top V\left(\tilde{\Lambda} - \Lambda\right)V^\top(\theta_{t+1} - \theta_t)\, dydz \\
&= \int_0^1 \int_0^1 y \sum_{j=1}^n \sum_{i=1}^n (\theta_{t+1} - \theta_t)^\top v_i \cdot (\theta_{t+1} - \theta_t)^\top v_j \cdot v_j^\top V\left(\tilde{\Lambda} - \Lambda\right)V^\top v_i dydz \\
&= \int_0^1 \int_0^1 y \cdot \sum_{i=1}^n \left((\theta_{t+1} - \theta_t)^\top v_i\right)^2 \cdot \left(\tilde{\lambda}_i - \lambda_i\right) dydz \\
&\leq \sum_{i=1}^n L_t^i \cdot \left|(\theta_{t+1} - \theta_t)^\top v_i\right|^3 \cdot \int_0^1 \int_0^1 y \cdot yz \cdot dydz \\
&= \sum_{i=1}^n \frac{L_t^i}{6} \cdot \left|(\theta_{t+1} - \theta_t)^\top v_i\right|^3
\end{aligned}
$$

As for the second term,

$$\begin{aligned}
&\int_0^1\int_0^1 y\left(\theta_{t+1}-\theta_t\right)^\top\left(\tilde{V}\tilde{\Lambda}\tilde{V}^\top-V\tilde{\Lambda}V^\top\right)\left(\theta_{t+1}-\theta_t\right)dydz\\
&=\int_0^1\int_0^1 y\left(\theta_{t+1}-\theta_t\right)^\top\left(\tilde{V}-V\right)\tilde{\Lambda}\left(\tilde{V}+V\right)^\top\left(\theta_{t+1}-\theta_t\right)dydz\\
&+\int_0^1\int_0^1 y\left(\theta_{t+1}-\theta_t\right)^\top\left(\left(\tilde{V}\tilde{\Lambda}V^\top\right)^\top-\tilde{V}\tilde{\Lambda}V^\top\right)\left(\theta_{t+1}-\theta_t\right)dydz\\
&=\int_0^1\int_0^1 y\left(\theta_{t+1}-\theta_t\right)^\top\left(\tilde{V}-V\right)\tilde{\Lambda}\left(\tilde{V}+V\right)^\top\left(\theta_{t+1}-\theta_t\right)dydz\\
&\le\int_0^1\int_0^1 y\cdot\left\|\tilde{V}-V\right\|_2\cdot\left\|\tilde{\Lambda}\right\|_2\cdot\left\|\tilde{V}+V\right\|_2\cdot\left\|\theta_{t+1}-\theta_t\right\|_2^2\cdot dydz\\
&\le\frac{1}{3}\cdot L_R\left\|\tilde{\Lambda}\right\|_2\cdot\left\|\theta_{t+1}-\theta_t\right\|_2^3\\
&=\frac{1}{3}\cdot L_R\left\|\tilde{\Lambda}\right\|_2\cdot\left\|\left(\theta_{t+1}-\theta_t\right)V^\top\right\|_2^3\\
&\le\frac{\sqrt{n}}{3}\cdot L_R\left\|\tilde{\Lambda}\right\|_2\cdot\left\|\left(\theta_{t+1}-\theta_t\right)V^\top\right\|_3^3\\
&=\frac{\sqrt{n}}{3}\cdot L_R\left\|\tilde{\Lambda}\right\|_2\cdot\sum_{i=1}^n\left|\left(\theta_{t+1}-\theta_t\right)V^\top\cdot e_i\right|^3\\
&=\sum_{i=1}^n\frac{\sqrt{n}}{3}\cdot L_R\left\|\tilde{\Lambda}\right\|_2\cdot\left|\left(\theta_{t+1}-\theta_t\right)v_i\right|^3
\end{aligned}$$

with

- the first 4 transfers similar to those in the proof of lemma P.4
- the third equality due to orthonormality
- the third inequality due to the L_p norms inequality
- e_i indicating the 1-hot vector with a 1 in the i-th entry

Putting it all together (and representing $\theta_{t+1}-\theta_t$ by its coordinate vector over the eigenbasis of $\mathcal{H}\left(\theta_t\right)$):

$$\begin{aligned}
f\left(\theta_{t+1}\right)-f\left(\theta_t\right)\le\sum_{i=1}^n\nabla f\left(\theta_t\right)^\top v_i\cdot\left(\theta_{t+1}-\theta_t\right)^\top v_i+\lambda_i\left(\theta_t\right)\cdot\left(\left(\theta_{t+1}-\theta_t\right)^\top v_i\right)^2\\
+\left(\frac{L_t^i}{6}+\frac{\sqrt{n}}{3}\cdot L_R\cdot L_\sigma\right)\cdot\left|\left(\theta_{t+1}-\theta_t\right)v_i\right|^3
\end{aligned}$$

□

To understand the relationship between our assumption 2 and the more standard equation 2, we further prove that the combination of assumption 2 and a bounded spectrum assumption will be no weaker than equation 2:

Lemma P.3. *Let $A\in\mathbb{R}^{n\times n}, v\in\mathbb{R}^n$. Then $v^\top\left(A^\top-A\right)v=0$.*

Proof.

$$v^\top\left(A^\top-A\right)v=\left(v^\top\left(A^\top-A\right)v\right)^\top=v^\top\left(A-A^\top\right)v=-v^\top\left(A^\top-A\right)v$$

□

Theorem P.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function satisfying assumptions 1 and 2. Then equation 2 is satisfied.*

Proof. Let $p \in \mathbb{R}^n$ s.t. $\|p\|_2 = 1$.

$$\begin{aligned}
\left|p^\top \left(\mathcal{H}\left(\theta\right) - \mathcal{H}\left(\varphi\right)\right) p\right| &= \left|p^\top \left(V\Lambda V^\top - \tilde{V}\tilde{\Lambda}\tilde{V}^\top\right) p\right| \\
&\leq \left|p^\top \left(V\Lambda V^\top - \tilde{V}\Lambda\tilde{V}^\top\right) p\right| + \left|p^\top \tilde{V}\left(\Lambda - \tilde{\Lambda}\right)\tilde{V}^\top p\right| \\
&= \left|p^\top \left(V - \tilde{V}\right)\Lambda\left(V + \tilde{V}\right)^\top p\right| + \left|p^\top \tilde{V}\left(\Lambda - \tilde{\Lambda}\right)\tilde{V}^\top p\right| \\
&\leq \left\|V - \tilde{V}\right\|_2 \cdot \|\Lambda\|_2 \cdot \left\|V + \tilde{V}\right\|_2 \cdot \|p\|_2^2 + \left\|\tilde{V}^\top p\right\|_2^2 \cdot \left\|\Lambda - \tilde{\Lambda}\right\|_2 \\
&\leq \left(2L_R \cdot \sup_{\theta' \in \mathbb{R}^n, i \in [n]} \lambda_i\left(\theta'\right)\right) \cdot \|\theta - \varphi\|_2 + \max_i L^i \cdot \|\theta - \varphi\|_2 \\
&\leq \left(2L_H \cdot L_\sigma + L_H\right) \cdot \|\theta - \varphi\|_2
\end{aligned}$$

with

- the first inequality due to the triangle inequality
- the second equality due to lemma P.3
- the second inequality due to the Cauchy-Schwartz inequality
- the third inequality due to the triangle inequality (for $\left\|V + \tilde{V}\right\|_2$), and the fact that an orthonormal matrix's eigenvalues' moduli are all 1.

□

P.3 Theorem D.1: Diminishing regret gains

Before we can prove theorem D.1, we need to upper bound equation 7.

Lemma P.5 (Minmax stepsize bound)**.** *If $\lambda_i \geq 0$ then*

$$\Delta\theta_t^{*\top} v_i = \mathcal{O}\left(\sqrt{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right)$$

If $\lambda_i < 0$ then

$$\Delta\theta_t^{*\top} v_i = \mathcal{O}\left(\left|\lambda_i\right|\right)$$

Proof. For i s.t. $0 \leq \lambda_i \leq \sqrt{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}$ we use corollary P.1.1 with $x = 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|$ to obtain

$$\begin{aligned}
&\Delta\theta_t^{*\top} v_i \\
&= \frac{\sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - \lambda_i}{L_t^i} \\
&\leq \sqrt{2} \cdot \sqrt{\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i}} + \frac{\frac{\lambda_i^2}{2} \cdot \frac{1}{\sqrt{2L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}} - \lambda_i}{L_t^i} \\
&\leq \sqrt{2} \cdot \sqrt{\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i}} + \frac{1}{2\sqrt{2}} \cdot \sqrt{\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i}} \\
&= \frac{5}{2\sqrt{2}} \cdot \sqrt{\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i}}
\end{aligned}$$

For i s.t. $\lambda_i > \sqrt{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}$ we use corollary P.1.1 with $x = \lambda_i^2$ to obtain

$$\begin{aligned}
\Delta\theta_t^{*\top} v_i &\leq \frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i} \\
&\leq \frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\sqrt{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}} = \sqrt{\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i}}
\end{aligned}$$

For i s.t. $\lambda_i < 0$, we again use corollary P.1.1 with $x = \lambda_i^2$ to obtain

$$\begin{aligned}
&\Delta\theta_t^{*\top} v_i \\
&= \frac{2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i \left(\sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - |\lambda_i|\right)} \\
&\leq \frac{2\left|\lambda_i\right|}{L_t^i} = \mathcal{O}\left(|\lambda_i|\right)
\end{aligned}$$

□

We are now ready to prove theorem D.1.

Theorem. *Diminishing regret gains Let f be a function with Lipschitz-continuous Hessian. After t iterations, algorithm* ELMO *satisfies*

$$f\left(\theta_0\right) - f\left(\theta_t\right) = \mathcal{O}\left(\log t\right) \tag{23}$$

Proof. Cartis et al. (2012a) give $\left|\nabla f\left(\theta_t\right)^\top v_i\right| = \mathcal{O}\left(\frac{1}{t^{\frac{2}{3}}}\right)$ and $\forall_{i:\lambda_i<0} : |\lambda_i| = \mathcal{O}\left(\frac{1}{\sqrt[3]{t}}\right)$ for the ARC optimization algorithm, of which algorithm ELMO is a special case (each eigenspace of ELMO is the 1-dimensional case where ARC perfectly estimates the Hessian Lipschitz parameter).

Making use of lemma P.5 and noting that $m_t^i\left(\Delta\theta_t^{*\top} v_i\right) \leq 0$ by equation 19:

$$\left|m_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right| = \left|\nabla f\left(\theta_t\right)^\top v_i\right| \cdot \Delta\theta_t^{*\top} v_i + \frac{-\lambda_i}{2} \cdot \left(\Delta\theta_t^{*\top} v_i\right)^2 + \frac{L_t^i}{6} \cdot \left(\Delta\theta_t^{*\top} v_i\right)^3$$

For i s.t. $\lambda_i \geq 0$:

$$\begin{aligned}
&\leq \left|\nabla f\left(\theta_t\right)^\top v_i\right| \cdot \mathcal{O}\left(\sqrt{\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i}}\right) + \mathcal{O}\left(\sqrt{\frac{\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{L_t^i}}\right)^3 \\
&= \mathcal{O}\left(\left|\nabla f\left(\theta_t\right)^\top v_i\right|^{1.5}\right) = \mathcal{O}\left(\left(\frac{1}{t^{\frac{2}{3}}}\right)^{1.5}\right) = \mathcal{O}\left(\frac{1}{t}\right)
\end{aligned}$$

For i s.t. $\lambda_i < 0$:

$$\begin{aligned}
&\leq \left|\nabla f\left(\theta_t\right)^\top v_i\right| \cdot \mathcal{O}\left(|\lambda_i|\right) + |\lambda_i| \cdot \left(\mathcal{O}\left(|\lambda_i|\right)\right)^2 + \mathcal{O}\left(|\lambda_i|\right)^3 \\
&= \mathcal{O}\left(\left|\nabla f\left(\theta_t\right)^\top v_i\right| \cdot |\lambda_i| + |\lambda_i|^3\right) \\
&= \mathcal{O}\left(\frac{1}{t^{\frac{2}{3}}} \cdot \frac{1}{\sqrt[3]{t}} + \frac{1}{t}\right) = \mathcal{O}\left(\frac{1}{t}\right)
\end{aligned}$$

Finally, we have

$$\begin{aligned}
f\left(\theta_0\right) - f\left(\theta_T\right) &= \sum_{t=0}^{T-1} f\left(\theta_t\right) - f\left(\theta_{t+1}\right) \\
&\leq \sum_{t=0}^{T-1} \left|m_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right| \\
&\leq \sum_{t=1}^{T-1} \mathcal{O}\left(\frac{1}{t}\right) \\
&\leq \mathcal{O}\left(\log t + \gamma + \frac{1}{2t}\right) \\
&= \mathcal{O}\left(\log t\right)
\end{aligned}$$

with $\gamma \approx 0.57721$ as the Euler-Mascheroni constant and Young (1991) for the last inequality. □

P.4 THEOREM D.2

Theorem.

$$\left|m_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right| \leq 5\left|M_t^i\left(\Delta\theta_t^{*\top} v_i\right)\right|$$

Proof. Due to equation 19, we have $\frac{m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top}v_i\right)} = \left|\frac{m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top}v_i\right)}\right|$. Now:

$$
\begin{aligned}
&\frac{m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top}v_i\right)} \\
&= \frac{\nabla f\left(\theta_t\right)^\top v_i \cdot \Delta\theta_t^{*\top}v_i + \frac{\lambda_i}{2}\left(\Delta\theta_t^{*\top}v_i\right)^2 - \frac{L_t^i}{6}\cdot\left|\Delta\theta_t^{*\top}v_i\right|^3}{\nabla f\left(\theta_t\right)^\top v_i \cdot \Delta\theta_t^{*\top}v_i + \frac{\lambda_i}{2}\left(\Delta\theta_t^{*\top}v_i\right)^2 + \frac{L_t^i}{6}\cdot\left|\Delta\theta_t^{*\top}v_i\right|^3} \\
&= \frac{-\left|\nabla f\left(\theta_t\right)^\top v_i\right| + \frac{\lambda_i}{2}\frac{\sqrt{\lambda_i^2+2L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}-\lambda_i}{L_t^i} - \frac{L_t^i}{6}\cdot\left(\frac{\sqrt{\lambda_i^2+2L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}-\lambda_i}{L_t^i}\right)^2}{-\left|\nabla f\left(\theta_t\right)^\top v_i\right| + \frac{\lambda_i}{2}\frac{\sqrt{\lambda_i^2+2L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}-\lambda_i}{L_t^i} + \frac{L_t^i}{6}\cdot\left(\frac{\sqrt{\lambda_i^2+2L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}-\lambda_i}{L_t^i}\right)^2} \\
&= \frac{5\left(\lambda_i\sqrt{\lambda_i^2+2L_t^i\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i^2\right) - 8L_t^i\left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\left(\lambda_i\sqrt{\lambda_i^2+2L_t^i\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i^2\right) - 4L_t^i\left|\nabla f\left(\theta_t\right)^\top v_i\right|}
\end{aligned}
$$

If $\lambda_i = 0$, then

$$\frac{m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top}v_i\right)} = 2$$

If $\lambda_i > 0$, then

$$\frac{m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top}v_i\right)} = \frac{5\frac{\sqrt{1+2\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}-1}{\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}} - 8}{\frac{\sqrt{1+2\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}-1}{\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}} - 4} \leq \lim_{x\to\infty}\frac{5\frac{\sqrt{1+2x}-1}{x}-8}{\frac{\sqrt{1+2x}-1}{x}-4} = 2$$

due to the monotonic increasing nature of $\psi_5 : \mathbb{R}^+ \to \mathbb{R}, \psi_5\left(x\right) = \frac{5\frac{\sqrt{1+2x}-1}{x}-8}{\frac{\sqrt{1+2x}-1}{x}-4}$.

If $\lambda_i < 0$, then

$$\frac{m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top}v_i\right)} = \frac{5\frac{\sqrt{1+2\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}}+1}{\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}} + 8}{\frac{\sqrt{1+2\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}}+1}{\frac{L_t^i\left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}} + 4} \leq \lim_{x\to 0^+}\frac{5\frac{\sqrt{1+2x}+1}{x}+8}{\frac{\sqrt{1+2x}+1}{x}+4} = 5$$

due to the monotonic decreasing nature of $\psi_6 : \mathbb{R}^+ \to \mathbb{R}, \psi_6\left(x\right) = \frac{5\frac{\sqrt{1+2x}+1}{x}+8}{\frac{\sqrt{1+2x}+1}{x}+4}$. □

P.5 THEOREM C.1: WORST-CASE DESCENT RATE FOR ARBITRARY OPTIMIZERS

Theorem. *Relative Descent Rate*

- $$\left|\frac{M_t^i\left(\Delta\theta_t^{\top}v_i\right) - M_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top}v_i\right)}\right| = \Theta\left(\left|\Delta\Delta^i\theta_t'\right|^2\right)$$

- $$\left|\frac{m_t^i\left(\Delta\theta_t^{\top}v_i\right) - m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}{m_t^i\left(\Delta\theta_t^{*\top}v_i\right)}\right| = \Theta\left(\left|\Delta\Delta^i\theta_t'\right|\right) \tag{24}$$

Proof. For the first part of the lemma,

$$
\begin{aligned}
&\left|\frac{M_t^i\left(\Delta\theta_t^\top v_i\right)-M_t^i\left(\Delta\theta_t^{*\top} v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top} v_i\right)}\right| \\
&=\frac{\nabla f\left(\theta_t\right)^\top \cdot v_i \cdot\left(\left(\theta_{t+1}-\theta_t\right)^\top v_i-\Delta\theta_t^{*\top} v_i\right)}{\Delta\theta_t^{*\top} v_i \cdot \nabla f\left(\theta_t\right)^\top v_i+\frac{1}{2}\lambda_i\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^3} \\
&+\frac{\frac{1}{2}\lambda_i \cdot\left(\left(\left(\theta_{t+1}-\theta_t\right)^\top v_i\right)^2-\left(\Delta\theta_t^{*\top} v_i\right)^2\right)}{\Delta\theta_t^{*\top} v_i \cdot \nabla f\left(\theta_t\right)^\top v_i+\frac{1}{2}\lambda_i\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^3} \\
&+\frac{\frac{L_t^i}{6} \cdot\left(\left(\left(\theta_{t+1}-\theta_t\right)^\top \cdot v_i\right)^3-\left(\Delta\theta_t^{*\top} v_i\right)^3\right)}{\Delta\theta_t^{*\top} v_i \cdot \nabla f\left(\theta_t\right)^\top v_i+\frac{1}{2}\lambda_i\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^3} \\
&=\Delta\Delta^i\theta_t^{\prime}\left(\frac{\frac{L_t^i}{6} \cdot\left(\Delta\Delta^i\theta_t^{\prime 2}+2\Delta\Delta^i\theta_t^{\prime}+3\right) \cdot\left(\Delta\theta_t^{*\top} v_i\right)^2}{\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{1}{2}\lambda_i\Delta\theta_t^{*\top} v_i-\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right. \\
&+\frac{\lambda_i \cdot\left(\frac{1}{2}\Delta\Delta^i\theta_t^{\prime}+1\right) \cdot \Delta\theta_t^{*\top} v_i}{\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{1}{2}\lambda_i\Delta\theta_t^{*\top} v_i-\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \\
&\left.-\frac{\left|\nabla f\left(\theta_t\right)^\top \cdot v_i\right|}{\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{1}{2}\lambda_i\Delta\theta_t^{*\top} v_i-\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right) \\
&=\Delta\Delta^i\theta_t^{\prime}\left(\frac{\frac{1}{6} \cdot\left(\Delta\Delta^i\theta_t^{\prime 2}+2\Delta\Delta^i\theta_t^{\prime}+3\right) \cdot \frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}}{\frac{1}{6}\frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}+\frac{1}{2}\lambda_i\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}-\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right. \\
&+\frac{\lambda_i \cdot\left(\frac{1}{2}\Delta\Delta^i\theta_t^{\prime}+1\right) \cdot\left(\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}\right)}{\frac{1}{6}\frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}+\frac{1}{2}\lambda_i\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}-\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \\
&\left.-\frac{\left|\nabla f\left(\theta_t\right)^\top \cdot v_i\right|}{\frac{1}{6}\frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}+\frac{1}{2}\lambda_i\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}-\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right) \\
&=\Delta\Delta^i\theta_t^{\prime} \cdot\left(\frac{\Delta\Delta^i\theta_t^{\prime}+2\Delta\Delta^i\theta_t^{\prime 2}}{\lambda_i\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i^2-4L_t^i\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \cdot \lambda_i^2\right. \\
&+\frac{2\left(\Delta\Delta^i\theta_t^{\prime 2}+2\Delta\Delta^i\theta_t^{\prime}\right)}{\lambda_i\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i^2-4L_t^i\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \cdot L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right| \\
&\left.-\frac{\Delta\Delta^i\theta_t^{\prime}+2\Delta\Delta^i\theta_t^{\prime 2}}{\lambda_i\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i^2-4L_t^i\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \cdot \lambda_i\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right)
\end{aligned}
$$

If $\lambda_i = 0$:

$$= -\Delta\Delta^i {\theta'_t}^2 \cdot \left(1 + \frac{1}{2}\Delta\Delta^i \theta'^2_t\right)$$

If $\lambda_i > 0$:

$$\begin{aligned}
&= \Delta\Delta^i \theta'^2_t \cdot \frac{1}{\frac{1}{\sqrt{1+2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}+1} - 2} \\
&\cdot \left(\left(\Delta\Delta^i \theta'_t + 2\right) - \left(1 + 2\Delta\Delta^i \theta'_t\right) \cdot \frac{1}{1 + \sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}} \right) \\
&= \Delta\Delta^i \theta'^2_t \cdot \left(\Delta\Delta^i \theta'_t - 1 - \sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i^2}} \cdot \left(\Delta\Delta^i \theta'_t + 2\right) \right) \\
&\cdot \frac{1}{1 + 2\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}} \\
&= \Delta\Delta^i \theta'^3_t \cdot \frac{1}{1 + 2\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}} \\
&- \Delta\Delta^i \theta'^2_t \cdot \left(1 + \frac{1}{2} \left(1 - \frac{1}{1 + 2\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}} \right) \cdot \Delta\Delta^i \theta'_t \right) \\
&= \left(\frac{3}{2} \frac{1}{1 + 2\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}} - \frac{1}{2} \right) \cdot \Delta\Delta^i \theta'^3_t - \Delta\Delta^i \theta'^2_t
\end{aligned}$$

Proving that

$$\frac{1}{1 + 2\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2}}} \in \left(0, \frac{1}{3}\right]$$

would conclude the proof for this case. This is easily proven, by noting that

$$\psi_1 : \mathbb{R}^+ \to \mathbb{R}, \psi_1\left(x\right) = \left(1 + 2\sqrt{1 + 2x}\right)^{-1}$$

is monotonic and satisfies

$$\lim_{x \to 0^+} \psi_1\left(x\right) = \frac{1}{3}$$

$$\lim_{x \to \infty} \psi_1\left(x\right) = 0$$

If, on the other hand, $\lambda_i < 0$:

$$= -\Delta\Delta^i\theta_t'^2 \cdot \left(\left(1 + 2\Delta\Delta^i\theta_t'\right) - \frac{3}{2}\Delta\Delta^i\theta_t' \cdot \frac{\frac{L_t^i \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}}{\frac{L_t^i \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2} + \frac{1}{4}\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}} + \frac{1}{4}} \right)$$

$$= \left(\frac{3}{2} \cdot \frac{\frac{L_t^i \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}}{\frac{L_t^i \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2} + \frac{1}{4}\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}} + \frac{1}{4}} - 2 \right) \Delta\Delta^i\theta_t'^3 - \Delta\Delta^i\theta_t'^2$$

Proving that

$$\frac{\frac{L_t^i \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}}{\frac{L_t^i \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2} + \frac{1}{4}\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{|\lambda_i|^2}} + \frac{1}{4}} \in [0, 1)$$

would conclude the proof for this case as well. This is easily proven, by noting that

$$\psi_2 : \mathbb{R}^+ \to \mathbb{R}, \psi_2(x) = \frac{x}{x + \frac{1}{4}\sqrt{1 + 2x} + \frac{1}{4}}$$

is monotonic and satisfies

$$\lim_{x \to 0^+} \psi_2(x) = 0$$

$$\lim_{x \to \infty} \psi_2(x) = 1$$

For the second part of the lemma,

$$
\begin{aligned}
&\left|\frac{m_t^i\left(\Delta\theta_t^\top v_i\right)-m_t^i\left(\Delta\theta_t^{*\top} v_i\right)}{m_t^i\left(\Delta\theta_t^{*\top} v_i\right)}\right| \\
&=\frac{\nabla f\left(\theta_t\right)^\top \cdot v_i \cdot\left(\left(\theta_{t+1}-\theta_t\right)^\top v_i-\Delta\theta_t^{*\top} v_i\right)}{\Delta\theta_t^{*\top} v_i \cdot \nabla f\left(\theta_t\right)^\top v_i+\frac{1}{2}\lambda_i\left(\Delta\theta_t^{*\top} v_i\right)^2-\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^3} \\
&+\frac{\frac{1}{2}\lambda_i \cdot\left(\left(\left(\theta_{t+1}-\theta_t\right)^\top v_i\right)^2-\left(\Delta\theta_t^{*\top} v_i\right)^2\right)}{\Delta\theta_t^{*\top} v_i \cdot \nabla f\left(\theta_t\right)^\top v_i+\frac{1}{2}\lambda_i\left(\Delta\theta_t^{*\top} v_i\right)^2-\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^3} \\
&-\frac{\frac{L_t^i}{6} \cdot\left(\left(\left(\theta_{t+1}-\theta_t\right)^\top \cdot v_i\right)^3-\left(\Delta\theta_t^{*\top} v_i\right)^3\right)}{\Delta\theta_t^{*\top} v_i \cdot \nabla f\left(\theta_t\right)^\top v_i+\frac{1}{2}\lambda_i\left(\Delta\theta_t^{*\top} v_i\right)^2-\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^3} \\
&=\Delta\Delta^i\theta_t'\left(\frac{\frac{-L_t^i}{6} \cdot\left(\Delta\Delta^i\theta_t'^2+2\Delta\Delta^i\theta_t'+3\right)}{-\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{1}{2}\lambda_i\Delta\theta_t^{*\top} v_i-\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \cdot\left(\Delta\theta_t^{*\top} v_i\right)^2\right. \\
&+\frac{\lambda_i \cdot\left(\frac{1}{2}\Delta\Delta^i\theta_t'+1\right)}{-\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{1}{2}\lambda_i\Delta\theta_t^{*\top} v_i-\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \cdot \Delta\theta_t^{*\top} v_i \\
&\left.-\frac{\left|\nabla f\left(\theta_t\right)^\top \cdot v_i\right|}{-\frac{L_t^i}{6}\left(\Delta\theta_t^{*\top} v_i\right)^2+\frac{1}{2}\lambda_i\Delta\theta_t^{*\top} v_i-\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right) \\
&=\Delta\Delta^i\theta_t'\left(\frac{-\frac{1}{6} \cdot\left(\Delta\Delta^i\theta_t'^2+2\Delta\Delta^i\theta_t'+3\right) \cdot \frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}}{-\frac{1}{6}\frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}+\frac{1}{2}\lambda_i\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}-\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right. \\
&+\frac{\lambda_i \cdot\left(\frac{1}{2}\Delta\Delta^i\theta_t'+1\right) \cdot\left(\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}\right)}{-\frac{1}{6}\frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}+\frac{1}{2}\lambda_i\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}-\left|\nabla f\left(\theta_t\right)^\top v_i\right|} \\
&\left.-\frac{\left|\nabla f\left(\theta_t\right)^\top \cdot v_i\right|}{-\frac{1}{6}\frac{\left(\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i\right)^2}{L_t^i}+\frac{1}{2}\lambda_i\frac{\sqrt{\lambda_i^2+2L_t^i \cdot\left|\nabla f\left(\theta_t\right)^\top v_i\right|}-\lambda_i}{L_t^i}-\left|\nabla f\left(\theta_t\right)^\top v_i\right|}\right)
\end{aligned}
$$

$$= \Delta\Delta^i\theta'_t \left(\frac{12\lambda_i \cdot \sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - 12\lambda_i^2 - 12L_t^i \left|\nabla f\left(\theta_t\right)^\top \cdot v_i\right|}{5\lambda_i \sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - 5\lambda_i^2 - 8L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} \right.$$

$$+ \frac{7\lambda_i \cdot \left(\sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - \lambda_i\right) - 4L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{5\lambda_i \sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - 5\lambda_i^2 - 8L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} \cdot \Delta\Delta^i\theta'_t$$

$$\left. + \frac{2\lambda_i \sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - 2\lambda_i^2 - 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{5\lambda_i \sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - 5\lambda_i^2 - 8L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} \cdot \Delta\Delta^i{\theta'_t}^2 \right)$$

Noting the common structure of each of the coefficients of $\Delta\Delta^i{\theta'_t}^1, \Delta\Delta^i{\theta'_t}^2, \Delta\Delta^i{\theta'_t}^3$, we prove the following to bound all three via appropriate settings of $a, b \in \{2, 4, 7, 12\}$:

If $\lambda_i > 0$:

$$\lim_{\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i^2} \to \mathcal{L}} \frac{a\lambda_i \cdot \left(\sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - \lambda_i\right) - bL_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{5\lambda_i \sqrt{\lambda_i^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} - 5\lambda_i^2 - 8L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}$$

$$= \lim_{\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i^2} \to \mathcal{L}} \frac{a\frac{2}{\sqrt{1+2\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i^2}}+1} - b}{5\frac{2}{\sqrt{1+2\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i^2}}+1} - 8}$$

$$= \begin{cases} \frac{b-a}{3} & \mathcal{L} = 0^+ \\ \frac{b}{8} & \mathcal{L} = \infty \end{cases}$$

If $\lambda_i \leq 0$:

$$\lim_{\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i^2} \to \mathcal{L}} \frac{a\left|\lambda_i\right| \cdot \left(\sqrt{\left|\lambda_i\right|^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} + \left|\lambda_i\right|\right) + bL_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{5\left|\lambda_i\right| \left(\sqrt{\left|\lambda_i\right|^2 + 2L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|} + \left|\lambda_i\right|\right) + 8L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}$$

$$= \lim_{\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\lambda_i^2} \to \mathcal{L}} \frac{a\left(\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\left|\lambda_i\right|^2}} + 1\right) + b\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\left|\lambda_i\right|^2}}{5\left(\sqrt{1 + 2\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\left|\lambda_i\right|^2}} + 1\right) + 8\frac{L_t^i \cdot \left|\nabla f\left(\theta_t\right)^\top v_i\right|}{\left|\lambda_i\right|^2}}$$

$$= \begin{cases} \frac{a}{5} & \mathcal{L} = 0^+ \\ \frac{b}{8} & \mathcal{L} = \infty \end{cases}$$

Analogously to the first case, and due to the monotonic natures (for all $a, b \in \mathbb{R}$) of

$$\psi_3 : \mathbb{R}^+ \to \mathbb{R}, \psi_3\left(x\right) = \frac{a\frac{2}{\sqrt{1+2x}+1} - b}{5\frac{2}{\sqrt{1+2x}+1} - 8}$$

and

$$\psi_4 : \mathbb{R}^+ \to \mathbb{R}, \psi_4\left(x\right) = \frac{a\left(\sqrt{1+2x}+1\right) + bx}{5\left(\sqrt{1+2x}+1\right) + 8x}$$

the term in the parentheses is bounded, thus we may conclude our proof of the lemma. □

Remark. *Note that when* $\lambda_i > 0$, $\frac{L_t^i \cdot \left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2} \to 0^+$, *the coefficients of* $\Delta\Delta^i{\theta'_t}^3, \Delta\Delta^i{\theta'_t}^1$ *shrink to 0 (since* $a = b$ *for those cases), so that* $\left|\frac{m_t^i\left(\Delta\theta_t^\top v_i\right) - m_t^i\left(\Delta\theta_t^{*\top} v_i\right)}{m_t^i\left(\Delta\theta_t^{*\top} v_i\right)}\right| = \Theta\left(\Delta\Delta^i{\theta'_t}^2\right)$

We are now ready to prove theorem C.1.

Theorem. *Worst-case descent rate for arbitrary optimizers*

Let $f : \mathbb{R}^n \to \mathbb{R}$ *a twice-differentiable function satisfying assumptions 1 and 2, and let* $\Delta\theta_t$ *satisfy* $M_t^i\left(\Delta\theta_t^\top v_i\right) \leq 0$. *Then*

- $$\left|\frac{M_t^i\left(\Delta\theta_t^\top v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top} v_i\right)}\right| = \Theta\left(1 + \left|\Delta\Delta^i\theta'_t\right|^2\right)$$

- $$\left|\frac{m_t^i\left(\Delta\theta_t^\top v_i\right)}{m_t^i\left(\Delta\theta_t^{*\top} v_i\right)}\right| = \Theta\left(1 + \left|\Delta\Delta^i\theta'_t\right|^p\right) \tag{25}$$

  with $p = \begin{cases} 2 & \lambda_i > 0 \text{ and } \frac{\left|\nabla f(\theta_t)^\top v_i\right|}{\lambda_i^2} = 0 \\ 1 & else \end{cases}$.

Proof. Proof is immediate from lemma P.5, because we have

$$\frac{M_t^i\left(\Delta\theta_t^\top v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top} v_i\right)} = 1 + \frac{M_t^i\left(\Delta\theta_t^\top v_i\right) - M_t^i\left(\Delta\theta_t^{*\top} v_i\right)}{M_t^i\left(\Delta\theta_t^{*\top} v_i\right)}$$

and similarly for $m_t^i\left(\Delta\theta_t^\top v_i\right)$. □

Q A STOCHASTIC CONVERGENCE RATE GUARANTEE FOR AMOS

This appendix establishes a non-asymptotic convergence rate for AMOS, under stochastic gradient and Hessian oracles, in a form directly comparable to the standard stochastic first-order baseline. The analysis is built on Assumption 2 rather than on the coarser equation (2): a single global Hessian-Lipschitz constant forces every direction to pay for the fastest-drifting direction in the model, which is precisely the pessimism the per-eigenspace refinement was introduced to remove, and an analysis conducted with it cannot express AMOS's advantage at all.

The guarantee turns on a single quantity — the largest Hessian-Lipschitz parameter among the eigenspaces assigned to the second-order optimizer, written here $L_2^{\max}$.

Q.1 THE ALGORITHM, AS ANALYZED

Time is divided into epochs. Epoch m begins at a refresh iteration τ_m and lasts at most k iterations, where k is the spacing of I_{lpshz} ($k = 10$ in the experiments of Appendix L). At each refresh:

1. A stochastic Hessian estimate $\widehat{H}_m$ of $H(\theta_{\tau_m})$ is formed ('OPT2.hessian', Algorithm `AMOS` line 4) and eigendecomposed, yielding an orthonormal basis $\{v_i\}_{i=1}^n$ and estimated curvatures $\hat{h}_i$. This basis is held fixed for the whole epoch.
2. Lipschitz estimates $\widehat{L}_i$ are updated by any rule depending only on data available before the current iteration. The exponential moving average of eigenvalue slopes in Algorithm 2 line 5 is such a rule, and no accuracy is required of it for the theorem to hold.
3. Directions are partitioned by thresholding the drift estimate: $S_1 = \{i : \widehat{L}_i > L_{\text{thresh}}\}$ is assigned to `OPT1` and $S_2 = \{i : \widehat{L}_i \leq L_{\text{thresh}}\}$ to 'OPT2'. The per-tensor granularity used in the implementation is the special case in which the partition is constant on tensor blocks.

4. Regularized curvatures $\bar{\lambda}_i$, $i \in S_2$, are set from $\widehat{H}_m$ subject to Assumption 6 below. Sophia's clipping $\bar{\lambda}_i = \max(\hat{h}_i, \rho)$ and ELMO's spectral regularization rule of section 3.2.1 with a floor both qualify.

Within the epoch, at iteration t a fresh stochastic gradient $\hat{g}_t$ is drawn, $\hat{g}_{t,i} := v_i^\top \hat{g}_t$, and

$$\theta_{t+1} = \theta_t + \Delta_t, \qquad \Delta_t = \underbrace{-\alpha \sum_{i \in S_1} \hat{g}_{t,i}\, v_i}_{\Delta_{t,1}} \underbrace{- \eta \sum_{i \in S_2} \frac{\hat{g}_{t,i}}{\bar{\lambda}_i}\, v_i}_{\Delta_{t,2}}. \tag{26}$$

Thus S_1 receives SGD at learning rate α and S_2 receives a damped regularized Newton step with damping $\eta \in (0, 1]$, the damping playing the role of 'OPT2''s learning rate. Write P_1, P_2 for the orthogonal projections onto the spans of $\{v_i\}_{i \in S_1}$ and $\{v_i\}_{i \in S_2}$, so that $\Delta_{t,b} = P_b \Delta_t$ and $\|\Delta_t\|^2 = \|\Delta_{t,1}\|^2 + \|\Delta_{t,2}\|^2$.

Q.2 Assumptions

Although some assumptions were introduced earlier, we state our assumptions for this section separately here because this stochastic, blockwise convergence analysis requires a more specialized set of assumptions, some of which refine or differ slightly from those used previously.

Throughout, $g_t := \nabla f(\theta_t)$, $H(\theta) := \nabla^2 f(\theta)$, $\Delta_0 := f(\theta_0) - f^*$, and $D_t := \|\theta_t - \theta_{\tau_m}\|$. Norms are Euclidean on vectors and spectral on matrices.

Assumption 3 (Regularity)**.** *f is twice differentiable, bounded below by f^*, satisfies $\|H(\theta)\| \le L_\sigma$ as in Assumption 1, and $\|\nabla f(\theta)\| \le G$ along the iterates.*

Assumption 4 (Second-order block drift in the refresh basis)**.** *For each epoch, with $\{v_i\}$ the refresh basis in use, there exists a constant Γ_2 such that for every θ reachable within the epoch,*

$$\big\| P_2 \big(H(\theta) - H(\theta_{\tau_m}) \big) P_2 \big\| \le \Gamma_2 \, \|\theta - \theta_{\tau_m}\|. \tag{27}$$

Only the second-order block requires a drift assumption. The first-order block is analyzed through the true curvature along the trajectory, exactly as in a standard first-order analysis, because SGD applies no preconditioner and its step therefore never needs to be related to the stale estimate $\widehat{H}_m$. Every Hessian comparison made below is anchored at the refresh point θ_{τ_m}, never between two interior iterates, so the eigenbasis appearing in Lemma Q.1 is always the refresh basis that defines P_1 and P_2. Writing

$$L_2^{\max} := \max_{i \in S_2} \bar{L}_i$$

for the largest per-eigenspace eigenvalue-Lipschitz parameter assigned to the second-order optimizer, Lemma Q.1 gives

$$\Gamma_2 \le L_2^{\max} + 2 L_\sigma L_R, \tag{28}$$

with L_R the eigenvector-rotation parameter of Assumption 2. Since $\bar{L}_i \le L_H$ and $L_R \le L_H$, this is bounded by $L_H(1 + 2L_\sigma)$, so Assumption 4 is never stronger than equation (2). The content of the refinement is that $L_2^{\max}$, and hence Γ_2, may be smaller than L_H by many orders of magnitude, as section Q.7 documents.

Assumption 5 (Stochastic gradient oracle)**.** *$\hat{g}_t = g_t + \xi_t$ where, conditionally on everything computed before iteration t: $\mathbb{E}[\xi_t] = 0$; $\mathbb{E}[\xi_t \xi_t^\top] \preceq \Sigma$ for a fixed positive semi-definite Σ with $\operatorname{tr} \Sigma \le \sigma^2$; and $\|\xi_t\| \le \sigma_a$ with certainty. Per-block noise levels are $\sigma_b^2 := \sup_m \sum_{i \in S_b^{(m)}} v_i^\top \Sigma v_i$, so that $\sigma_1^2 + \sigma_2^2 \le \sigma^2$ within any epoch.*

Assumption 6 (Curvature oracle and preconditioner)**.** *$\|\widehat{H}_m - H(\theta_{\tau_m})\| \le \delta_H$; the refresh estimate is computed from data independent of the gradient minibatches drawn subsequently; and for $i \in S_2$,*

$$\rho \le \bar{\lambda}_i \le \bar{\lambda}_{\max} \qquad \text{and} \qquad \bar{\lambda}_i \ge \hat{h}_i.$$

The domination $\bar{\lambda}_i \ge \hat{h}_i$ is supplied by construction both by the regularization rule of section 3.2.1 and by Sophia's clipping, on the same estimate from which they are computed; no relation to the true Hessian is assumed beyond δ_H.

Assumption 7 (Drift-estimation accuracy)**.** $|\widehat{L}_i - \bar{L}_i| \le \delta_L$ *at refresh times.*

Remark 4. *Assumption 7 is used only to convert the realized $L_2^{\max}$ into an a priori quantity: combined with the routing rule of section Q.1 it gives*

$$L_2^{\max} \le L_{\text{thresh}} + \delta_L. \tag{29}$$

Notation 9 (Step-size budget and drift exposure)**.** Fix $s_{\max} > 0$ and require $\alpha \le s_{\max}$ and $\eta/\rho \le s_{\max}$, so that every per-direction step scale is at most $s_{\max}$. Define

$$\Pi := \tfrac{1}{2}\, k\, s_{\max}\, (G + \sigma_a),$$

half the furthest the parameters can travel between Hessian refreshes. Every drift constant acts through this lever arm.

Notation 10 (Cross-block drift)**.** $\Gamma_\times$ is defined as a bound on the cross-block drift. In Lemma Q.1 we prove the existence of this bound.

$$\|P_1\big(H(\theta) - H(\theta_{\tau_m})\big)P_2\| \le \Gamma_\times \|\theta - \theta_{\tau_m}\|$$

Notation 11 (Curvature constants)**.** Two quantities describe the true curvature seen by the first-order block along the trajectory:

$$\lambda_1^+ := \sup_t \big\|\big(P_1 H(\theta_t) P_1\big)_+\big\|, \qquad C_\times := \sup_{\theta \in B(\theta_t, 2\Pi)} \big\|P_1 H(\theta) P_2\big\|, \qquad \forall t \in [T].$$

with $B(X, r) = \{x : \|x - X\|_2 \le r\}$ indicating a ball of radius r around $X \in \mathbb{R}^n$.

λ_1^+ is the largest positive curvature ever present in the subspace assigned to `OPT1`, and $C_\times$ measures the curvature coupling between the two subspaces. Both are pure smoothness quantities, bounded by L_σ: since P_1, P_2 are orthogonal projections, $\|P_1 x\| \le \|x\|$ and $\|P_2 y\| \le \|y\|$ for all x, y, so for unit vectors x, y,

$$|x^\top P_1 H(\theta_t) P_2\, y| = |(P_1 x)^\top H(\theta_t)(P_2 y)| \le \|H(\theta_t)\| \cdot \|P_1 x\| \cdot \|P_2 y\| \le \|H(\theta_t)\| \le L_\sigma$$

by Fact 2 and Assumption 3, and taking the supremum over unit x, y gives $C_\times \le L_\sigma$; the bound for λ_1^+ is immediate from the same inequality with $P_2 = P_1$. Neither constant involves a Hessian-Lipschitz parameter. Since the partition is by eigenspaces of $\widehat{H}_m$, for which $P_1 \widehat{H}_m P_2 = 0$ exactly, $C_\times$ is in fact small: $C_\times \le \min\big(L_\sigma,\ \delta_H + 2\Pi\Gamma_\times\big)$. This is because for any iterate θ_t within epoch m,

$$\begin{aligned}\|P_1 H(\theta_t) P_2\| &\le \|P_1(H(\theta_t) - H(\theta_{\tau_m}))P_2\| + \|P_1(H(\theta_{\tau_m}) - \widehat{H}_m)P_2\| \\ &\le \Gamma_\times \|\theta_t - \theta_{\tau_m}\| + \delta_H \le \delta_H + 2\Pi\Gamma_\times,\end{aligned}$$

where the first inequality uses the orthogonality of the projections P and the triangle inequality, and the final inequality uses $\|\theta_t - \theta_{\tau_m}\| \le k s_{\max}(G + \sigma_a) = 2\Pi$. Together with the global smoothness bound $C_\times \le L_\sigma$, this gives

$$C_\times \le \min\big(L_\sigma,\ \delta_H + 2\Pi\Gamma_\times\big).$$

At the refresh point, this coupling vanishes when the partition aligns with the eigenspaces of the true Hessian; away from the refresh point, its growth is controlled by the cross-block drift term $2\Pi\Gamma_\times$.

Notation 12 (Effective block constants)**.** Define

$$\Theta_1 := \lambda_1^+ + C_\times, \qquad \Theta_2 := 1 + \frac{\delta_H + C_\times + 2\Pi\Gamma_2}{\rho}. \tag{30}$$

Substituting equation 28 makes the dependence of the second-order constant on the assigned drift explicit:

$$\Theta_2 \le 1 + \frac{\delta_H + C_\times + 2\Pi\big(L_2^{\max} + 2L_\sigma L_R\big)}{\rho}. \tag{31}$$

These two constants occupy the position that the smoothness constant occupies in a first-order analysis, and the asymmetry between them is the substance of the result. Θ_1 is a pure smoothness constant: it contains the curvature encountered by the first-order block and nothing else, no Hessian-Lipschitz parameter and no dependence on the refresh interval, so the first-order block is governed by precisely the quantity that governs SGD. Θ_2 contains the drift rate $L_2^{\max}$ of the directions handed to the second-order optimizer, and contains no curvature magnitude at all. Each block is therefore charged for exactly what its own optimizer's validity depends on: SGD on curvature, the second-order step on curvature drift. Sections Q.6 and Q.7 trace the consequences for the rate and for the comparison with SGD.

Q.3 Preliminaries

Fact 1 (Law of total expectation)**.** *For any quantity X determined by the run of the algorithm, $\mathbb{E}[X] = \mathbb{E}\big[\mathbb{E}[X \mid \mathcal{F}_t]\big]$, where conditioning on $\mathcal{F}_t$ means averaging over the fresh minibatch while holding fixed every quantity computed before it was drawn: the iterate θ_t, the basis $\{v_i\}$, the partition, and the $\bar\lambda_i$. This is the only probabilistic device used below; each conditional expectation is an average over a single minibatch with all other quantities treated as constants.*

Fact 2 (Quadratic and bilinear forms)**.** *For symmetric A and any x, y, $|x^\top A y| \le \|A\|\,\|x\|\,\|y\|$ (Cauchy–Schwarz inequality); and $A \preceq B$ implies $v^\top A v \le v^\top B v$.*

Lemma Q.1 (From Assumption 2 to the second-order drift bound)**.** *Let θ be reachable within epoch m, and let $H(\theta_{\tau_m}) = V\Lambda V^\top$ and $H(\theta) = \tilde V \tilde\Lambda \tilde V^\top$ be eigendecompositions, where $V = [v_1 \cdots v_n]$ is the refresh basis up to the $O(\delta_H)$ gap between $H(\theta_{\tau_m})$ and $\widehat H_m$. Under Assumption 2, equation 27 holds with $\Gamma_2 \le L_2^{\max} + 2L_\sigma L_R$, and the cross-block drift obeys $\|P_1(H(\theta) - H(\theta_{\tau_m}))P_2\| \le \Gamma_\times \|\theta - \theta_{\tau_m}\|$ with $\Gamma_\times \le 2L_\sigma L_R$.*

Proof. Split the drift into a spectral part and a rotation part:

$$H(\theta) - H(\theta_{\tau_m}) = \underbrace{V(\tilde\Lambda - \Lambda)V^\top}_{F} + \underbrace{\big(\tilde V\tilde\Lambda\tilde V^\top - V\tilde\Lambda V^\top\big)}_{E}.$$

F is diagonal in V with entries $\tilde\lambda_i - \lambda_i$, and the first bullet of Assumption 2 gives $|\tilde\lambda_i - \lambda_i| \le \bar L_i |(\theta - \theta_{\tau_m})^\top v_i| \le \bar L_i \|\theta - \theta_{\tau_m}\|$. Hence $\|P_2 F P_2\| \le L_2^{\max}\|\theta - \theta_{\tau_m}\|$: the second-order block's drift constant depends only on the drift of the directions assigned to that block. Moreover

$$P_1 F P_2 = 0$$

exactly, whatever the size of the drift. Indeed F is diagonal in precisely the basis from which P_1 and P_2 are built, so writing $F = \sum_k (\tilde\lambda_k - \lambda_k) v_k v_k^\top$, each term of $P_1 F P_2$ would require v_k to lie simultaneously in the range of P_1 and in the range of P_2; these ranges are spanned by disjoint subsets of the same orthonormal basis and are therefore orthogonal, so no such v_k exists. A purely spectral drift produces no coupling between the blocks at all, however fast the eigenvalues move.

For the rotation part, write $E = (\tilde V - V)\tilde\Lambda\tilde V^\top + V\tilde\Lambda(\tilde V - V)^\top$, so that by orthogonality of V and $\tilde V$ and the second bullet of Assumption 2, $\|E\| \le 2\|\tilde V - V\|\|\tilde\Lambda\| \le 2L_R L_\sigma \|\theta - \theta_{\tau_m}\|$. Since E is the only source of cross-block coupling, $\Gamma_\times \le 2L_\sigma L_R$; combining the two parts gives equation 28.

□

Two consequences follow. First, the per-eigenspace parameter $\bar L_i$ enters Γ_2 only for $i \in S_2$, so a fast-drifting direction routed to `OPT1` does not affect the second-order block's constant at all.

This isolation is exact rather than asymptotic, and it is what the routing rule purchases. Second, all coupling between blocks is charged to eigenvector rotation alone, since $\Gamma_\times$ carries a factor L_R and no eigenvalue-drift term whatsoever; if L_R is small then $C_\times$ is correspondingly small. The isolation holds across blocks but not within one: $L_2^{\max}$ is a maximum over the whole of S_2, so a single fast-drifting curvature direction misrouted into S_2 inflates Θ_2 for every direction sharing that block, which is the analytic cost of coarse partitioning and bears on the tensor-granularity question raised in Appendix P.

Q.4 BLOCKWISE DESCENT

The classical descent lemma charges the whole step at a single smoothness constant, and its cubic refinement [Nesterov and Polyak, 2006, Lemma 1; Dennis and Schnabel, 1983, Lemma 4.1.14] charges the whole step at the global rate L_H. Neither is adequate here, because the two blocks require different treatment: the first-order block needs only smoothness, while the second-order block needs its step to be compatible with the stale estimate $\widehat{H}_m$ and therefore pays for curvature drift. The following lemma treats them asymmetrically.

Lemma Q.2 (Blockwise descent)**.** *Under Assumptions 3, 4 and 6, for any iterate θ_t in epoch m and any Δ with $\theta_t + \Delta$ reachable within the epoch,*

$$f(\theta_t+\Delta) \le f(\theta_t) + \nabla f(\theta_t)^\top \Delta + \frac{\lambda_1^+}{2}\|\Delta_1\|^2 \\ + C_\times \|\Delta_1\|\|\Delta_2\| + \frac{1}{2}\Delta_2^\top \widehat{H}_m \Delta_2 + \frac{\delta_H}{2}\|\Delta_2\|^2 + \Big(\frac{D_t}{2} + \frac{\|\Delta\|}{6}\Big)\Gamma_2\|\Delta_2\|^2.$$

Proof. By the fundamental theorem of calculus applied twice, as in the second proof of Lemma 3.1,

$$f(\theta_t+\Delta) = f(\theta_t) + \nabla f(\theta_t)^\top \Delta + \int_0^1 (1-u)\,\Delta^\top H(\theta_t + u\Delta)\,\Delta\, du.$$

Decompose the integrand over the two blocks, writing $H_u := H(\theta_t + u\Delta)$:

$$\Delta^\top H_u \Delta = \Delta_1^\top H_u \Delta_1 + 2\Delta_1^\top H_u \Delta_2 + \Delta_2^\top H_u \Delta_2.$$

The three pieces are treated differently, according to what each block's optimizer actually requires.

For the first piece, Δ_1 lies in the range of P_1, so $\Delta_1^\top H_u \Delta_1 = \Delta_1^\top (P_1 H_u P_1)\Delta_1 \le \lambda_1^+ \|\Delta_1\|^2$ by the definition of λ_1^+ and Fact 2. No Hessian estimate and no drift constant appears: this is the ordinary smoothness bound of a first-order analysis, restricted to the subspace on which SGD operates.

For the cross piece, $2\Delta_1^\top H_u \Delta_2 = 2\Delta_1^\top (P_1 H_u P_2)\Delta_2 \le 2C_\times \|\Delta_1\|\|\Delta_2\|$, again by definition and Fact 2.

Only the third piece is referred to the stale estimate, because only there does the analysis need the domination $\bar{\lambda}_i \ge \hat{h}_i$ that justifies the second-order step. Adding and subtracting,

$$\Delta_2^\top H_u \Delta_2 = \Delta_2^\top \widehat{H}_m \Delta_2 + \Delta_2^\top \big(H_u - H(\theta_{\tau_m})\big)\Delta_2 + \Delta_2^\top \big(H(\theta_{\tau_m}) - \widehat{H}_m\big)\Delta_2,$$

where the middle term is anchored at the refresh point and, since Δ_2 lies in the range of P_2, is bounded using equation 27 by $\Gamma_2\|\theta_t + u\Delta - \theta_{\tau_m}\|\,\|\Delta_2\|^2 \le \Gamma_2(D_t + u\|\Delta\|)\|\Delta_2\|^2$, and the last term is at most $\delta_H\|\Delta_2\|^2$ by Assumption 6.

Integrating and using $\int_0^1 (1-u)\,du = \frac{1}{2}$ together with $\int_0^1 (1-u)(D_t + u\|\Delta\|)\,du = \frac{D_t}{2} + \frac{\|\Delta\|}{6}$ gives the stated bound. □

The asymmetry is the point. The first-order block is charged $\frac{\lambda_1^+}{2}\|\Delta_1\|^2$, exactly the term a standard analysis of SGD would produce with λ_1^+ in place of the global smoothness constant; it pays nothing for curvature drift, nothing for the staleness of the Hessian, and nothing for its estimation error,

because SGD uses none of that information. The second-order block is charged for all three, since its step is built from the stale estimate and is valid only to the extent that the estimate remains accurate over the epoch. A small Γ_2 multiplies $\|\Delta_2\|^2$, the long Newton step, which is affordable precisely because its coefficient is small; the global-L_H form charges $L_H\|\Delta_2\|^2$ instead, forcing the second-order step to be as conservative as the worst direction anywhere in the model.

Q.5 One-step descent

Fix an iteration t within epoch m and drop epoch indices. Write

$$\hat{Q} := \sum_{i\in S_2} \frac{\hat{g}_i^2}{\bar{\lambda}_i}, \qquad Q_t := \sum_{i\in S_2} \frac{(v_i^\top g_t)^2}{\bar{\lambda}_i}, \qquad N_2 := \sup_m \sum_{i\in S_2^{(m)}} \frac{v_i^\top \Sigma v_i}{\bar{\lambda}_i} \le \frac{\sigma_2^2}{\rho}.$$

The quantity Q_t is the curvature-normalized gradient energy on the second-order block, measuring each direction's squared gradient in units of its curvature; it is the progress measure that the block's own step decreases.

Lemma Q.3 (One-step expected descent)**.** *Under Assumptions 3–6, if*

$$\alpha \le \min\Big(\frac{1}{\Theta_1},\, s_{\max}\Big), \qquad \eta \le \min\Big(\frac{1}{\Theta_2},\, \rho\, s_{\max},\, 1\Big), \tag{32}$$

then for every t,

$$\mathbb{E}\big[f(\theta_{t+1})\big] - \mathbb{E}\big[f(\theta_t)\big] \le -\frac{\alpha}{2}\mathbb{E}\|P_1 g_t\|^2 - \frac{\eta}{2}\mathbb{E}[Q_t] + \frac{\Theta_1\alpha^2}{2}\sigma_1^2 + \frac{\Theta_2\eta^2}{2}N_2. \tag{33}$$

Proof. The proof proceeds in six steps.

Step 1 Lemma Q.2 supplies the descent inequality, already split by block and charging each for what its own optimizer requires:

$$f(\theta_{t+1}) \le f(\theta_t) + g^\top\Delta + \frac{\lambda_1^+}{2}\|\Delta_1\|^2 \tag{34}$$

$$+ C_\times\|\Delta_1\|\|\Delta_2\| + \frac{1}{2}\Delta_2^\top\widehat{H}\Delta_2 + \frac{\delta_H}{2}\|\Delta_2\|^2 + \Big(\frac{D_t}{2} + \frac{\|\Delta\|}{6}\Big)\Gamma_2\|\Delta_2\|^2 \tag{35}$$

Step 2 Because $\widehat{H}$ is exactly diagonal in the refresh basis, its quadratic form on the second-order block decouples across directions with no cross terms, and the domination $\hat{h}_i \le \bar{\lambda}_i$ of Assumption 6 applies per direction. With $\Delta_{2,i} = -\eta\hat{g}_i/\bar{\lambda}_i$ for $i \in S_2$,

$$\tfrac{1}{2}\Delta_2^\top\widehat{H}\Delta_2 = \tfrac{1}{2}\sum_{i\in S_2}\hat{h}_i\Delta_{2,i}^2 \le \tfrac{1}{2}\sum_{i\in S_2}\bar{\lambda}_i\cdot\frac{\eta^2\hat{g}_i^2}{\bar{\lambda}_i^2} = \frac{\eta^2}{2}\hat{Q}. \tag{36}$$

This is the mechanism supplied by the regularization rule of section 3.2.1 and by Sophia's clipping: since the preconditioner never understates the curvature of the model, the quadratic penalty of a second-order step never exceeds half its first-order gain, direction by direction and with no global constant. Where $\hat{h}_i < 0$ the left-hand side is negative and the bound is merely generous, discarding a strictly negative term in order to keep the statement uniform across blocks. Per-direction decoupling is available because AMOS eigendecomposes the estimate on which it acts.

Step 3 The first-order block requires no such treatment. With $\Delta_{1,i} = -\alpha\hat{g}_i$ for $i \in S_1$, simply $\|\Delta_1\| = \alpha\|P_1\hat{g}\|$, so its penalty in equation 35 is $\frac{\lambda_1^+}{2}\alpha^2\|P_1\hat{g}\|^2$, which is the penalty of a standard first-order analysis.

Step 4 The remaining terms are collected. Writing $\|\Delta_2\|^2 = \sum_{i\in S_2} \eta^2 \hat{g}_i^2/\bar{\lambda}_i^2 \le (\eta^2/\rho)\hat{Q}$, the estimation and drift penalties on the second-order block become $\frac{\delta_H}{2}(\eta^2/\rho)\hat{Q}$ and, using $\|\Delta\| \le s_{\max}(G+\sigma_a)$, $D_t \le (k-1)s_{\max}(G+\sigma_a)$ and $\frac{k-1}{2}+\frac{1}{6} \le \frac{k}{2}$,

$$\Big(\frac{D_t}{2} + \frac{\|\Delta\|}{6}\Big)\Gamma_2\|\Delta_2\|^2 \le \Pi\,\Gamma_2\,\frac{\eta^2}{\rho}\hat{Q}.$$

The cross term is split by Young's inequality, $2ab \le a^2+b^2$, contributing $\frac{C_\times}{2}\alpha^2\|P_1\hat{g}\|^2$ to the first block and $\frac{C_\times}{2}(\eta^2/\rho)\hat{Q}$ to the second. Substituting into equation 35 and reading off the definitions equation 30,

$$f(\theta_{t+1}) \le f(\theta_t) + g^\top\Delta + \frac{\Theta_1\alpha^2}{2}\|P_1\hat{g}\|^2 + \frac{\Theta_2\eta^2}{2}\hat{Q}. \tag{37}$$

The bound now has the shape of two independent first-order analyses, one per block, with Θ_1 and Θ_2 in the position of the smoothness constant. The first of these is a pure smoothness constant; it is only in the second that $L_2^{\max}$ enters, by equation 31, and thus governs the second-order block's admissible step size.

Step 5 Every quantity multiplying the step, namely α, η, the basis, the partition, the $\bar{\lambda}_i$ and g itself, was fixed before the minibatch was drawn, by Fact 1 and the independence clause of Assumption 6. Hence unbiasedness applies directly:

$$\mathbb{E}\big[g^\top\Delta \mid \mathcal{F}_t\big] = -\alpha\|P_1 g\|^2 - \eta Q_t.$$

For the quadratic terms, $\mathbb{E}[\hat{g}_i^2 \mid \mathcal{F}_t] = g_i^2 + \mathbb{E}[\xi_i^2 \mid \mathcal{F}_t] \le g_i^2 + v_i^\top\Sigma v_i$ by covariance domination, giving $\mathbb{E}\|P_1\hat{g}\|^2 \le \|P_1 g\|^2 + \sigma_1^2$ and $\mathbb{E}[\hat{Q}] \le Q_t + N_2$. Therefore

$$\mathbb{E}\big[f(\theta_{t+1}) \mid \mathcal{F}_t\big] \le f(\theta_t) - \alpha\Big(1-\frac{\Theta_1\alpha}{2}\Big)\|P_1 g\|^2 - \eta\Big(1-\frac{\Theta_2\eta}{2}\Big)Q_t + \frac{\Theta_1\alpha^2}{2}\sigma_1^2 + \frac{\Theta_2\eta^2}{2}N_2.$$

Step 6 The conditions equation 32 make each block's penalty at most half its own gain: $\alpha\Theta_1 \le 1$ and $\eta\Theta_2 \le 1$ turn the descent coefficients into $-\alpha/2$ and $-\eta/2$. Taking the outer expectation by Fact 1 yields equation 33. □

Q.6 Convergence rate

Theorem Q.4 (Preconditioned criticality)**.** *Under Assumptions 3–6 and the step-size conditions equation 32, for every horizon $T \ge 1$,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\Big[\alpha\|P_1\nabla f(\theta_t)\|^2 + \eta Q_t\Big] \le \frac{2\Delta_0}{T} + \Theta_1\alpha^2\sigma_1^2 + \Theta_2\eta^2 N_2. \tag{38}$$

Proof. Sum equation 33 over $t = 0, \dots, T-1$. The left-hand side telescopes to $\mathbb{E}f(\theta_T) - f(\theta_0) \ge -\Delta_0$. Rearranging and multiplying by $2/T$ gives equation 38. □

The left-hand side is the progress measure natural to a hybrid method: plain squared gradients on the first-order block and curvature-normalized ones on the second-order block. The transient term $2\Delta_0/T$ carries no curvature constant, so in the metric the algorithm optimizes the burn-in is curvature-free on the second-order block, which is the finite-horizon expression of the affine covariance of Newton-type methods.

Corollary Q.4.1 (Euclidean norm)**.** *Since $\alpha\|P_1 g\|^2 + \eta Q_t \ge \min(\alpha, \eta/\bar{\lambda}_{\max})\|\nabla f(\theta_t)\|^2$,*

$$\min_{0\le t<T}\mathbb{E}\|\nabla f(\theta_t)\|^2 \le \frac{1}{\min(\alpha, \eta/\bar{\lambda}_{\max})}\left[\frac{2\Delta_0}{T} + \Theta_1\alpha^2\sigma_1^2 + \Theta_2\eta^2 N_2\right]$$

.

Corollary Q.4.2 (Tuned step sizes)**.** *Take $\eta = \bar{\lambda}_{\max}\alpha$, so that both blocks contribute at scale α and $\min(\alpha, \eta/\bar{\lambda}_{\max}) = \alpha$. With*

$$V := \Theta_1\sigma_1^2 + \Theta_2\bar{\lambda}_{\max}^2 N_2, \qquad \bar{\alpha} := \min\left\{\frac{1}{\Theta_1},\ \frac{1}{\bar{\lambda}_{\max}\Theta_2},\ s_{\max},\ \frac{\rho\, s_{\max}}{\bar{\lambda}_{\max}}\right\},$$

the choice $\alpha = \min\left(\bar{\alpha}, \sqrt{2\Delta_0/(TV)}\right)$ gives, for every T,

$$\min_{0\le t<T} \mathbb{E}\|\nabla f(\theta_t)\|^2 \ \le\ 2\sqrt{\frac{2\Delta_0 V}{T}} \ +\ \frac{2\Delta_0}{\bar{\alpha}T}. \tag{39}$$

.

Both constants in equation 39 depend on $L_2^{\max}$ only through Θ_2, hence only through the second-order block: they degrade monotonically in $L_2^{\max}$ via V and via the term $1/(\bar{\lambda}_{\max}\Theta_2)$ in $\bar{\alpha}$, while the contribution $\Theta_1\sigma_1^2$ of the first-order block is unaffected by it. Making the dependence explicit by substituting equation 31 and retaining only the drift contribution,

$$\bar{\alpha} \le \frac{1}{\bar{\lambda}_{\max}\Theta_2} \le \frac{\rho}{\bar{\lambda}_{\max}\left(\rho + 2\Pi L_2^{\max}\right)}, \qquad V \ge \bar{\lambda}_{\max}^2 N_2\left(1 + \frac{2\Pi L_2^{\max}}{\rho}\right),$$

so the admissible step size falls and the asymptotic constant rises in proportion to the largest drift rate assigned to the second-order optimizer. Under Assumption 7 and the routing rule, $L_2^{\max} \le L_{\text{thresh}} + \delta_L$ by equation 29, so both quantities are controlled a priori by the threshold.

Corollary Q.4.3 (Threshold form)**.** *Suppose the routing rule and the refresh interval are chosen so that*

$$2\Pi\left(L_2^{\max} + 2L_\sigma L_R\right) \le \rho - \delta_H - C_\times. \tag{40}$$

Then $\Theta_2 \le 2$, the damping constraint in equation 32 is satisfied at $\eta = \frac{1}{2}$, and the per-direction learning rate in direction $i \in S_2$ is $1/(2\bar{\lambda}_i)$.

Under Assumption 7, equation 40 is implied by $L_{\text{thresh}} + \delta_L + 2L_\sigma L_R \le (\rho - \delta_H - C_\times)/(2\Pi)$, a condition on the tuning parameters alone.

Corollary Q.4.4 (First-order specialization)**.** *If $S_2 = \varnothing$ and $s_{max} \ge 1/\lambda_1^+$ then P_1 is the identity, $C_\times = 0$, and the update equation 26 is SGD; since $S_2 = \varnothing$ also gives $\sigma_1^2 = \sigma^2$ and $N_2 = 0$, the constant V of Corollary Q.4.2 collapses to $\lambda_1^+\sigma^2$. In that case $\Theta_1 = \lambda_1^+ = \sup_t \|(H(\theta_t))_+\|$ and equation 39 reads*

$$\min_{0\le t<T} \mathbb{E}\|\nabla f(\theta_t)\|^2 \le 2\sqrt{\frac{2\Delta_0\lambda_1^+\sigma^2}{T}} + \frac{2\lambda_1^+\Delta_0}{T},$$

which is the bound of Ghadimi & Lan (2013) with the smoothness constant L replaced by $\lambda_1^+ \le L_\sigma$, the largest positive curvature encountered along the trajectory. No Hessian-Lipschitz parameter, refresh interval or estimation error appears, since none of Assumptions 4, 6 or 7 is invoked when S_2 is empty. The guarantee for the first-order block is therefore the standard first-order guarantee, and the drift machinery is confined entirely to the block that uses second-order information.

Q.7 Comparison with first-order methods

For L-smooth f with gradient-noise variance σ^2, SGD with a tuned constant step size $\alpha \le 1/L$ satisfies

$$\min_{0\le t<T} \mathbb{E}\|\nabla f(\theta_t)\|^2 \le c_1\frac{L\Delta_0}{T} + c_2\sqrt{\frac{L\Delta_0\sigma^2}{T}} \tag{41}$$

for absolute constants c_1, c_2 (Ghadimi & Lan, 2013). Both terms carry the global constant L, the first because the admissible step is capped at $1/L$ and the second because the per-step variance penalty is $\frac{L}{2}\alpha^2\sigma^2$. With a bounded-variance stochastic gradient oracle the exponent $T^{-1/2}$ cannot be improved in general (Arjevani et al., 2019), so any stochastic guarantee for AMOS necessarily has the shape of equation 39 and the comparison rests entirely on the constants.

The admissible per-direction learning rates may be compared directly. That of SGD is $1/L$ in every direction, set by the stiffest direction anywhere in the model. That of AMOS on $i \in S_2$ is $\eta/\bar{\lambda}_i$ with $\eta \leq 1/\Theta_2$, and by equation 31 the constant Θ_2 contains the assigned drift rate $L_2^{\max}$, the estimation error δ_H and the block coupling $C_\times$, but no curvature magnitude of the assigned directions themselves. Whenever the assigned directions drift slowly enough for Corollary Q.4.3 to apply, the damping constraint is inactive at $\eta = \frac{1}{2}$ and the per-direction learning rate in direction i is $1/(2\bar{\lambda}_i)$ against SGD's $1/L$, an improvement by the factor $L/(2\bar{\lambda}_i)$, which is of the order of the condition number for the flat directions that dominate the spectrum.

This asymmetry is the analytic content of the argument of sections 3.2.1 and 6. A direction may be arbitrarily flat, indeed the flatter the better since the second-order step $|g_i|/\bar{\lambda}_i$ grows accordingly, and the guarantee is unaffected provided its curvature is not moving. Conversely a fast-drifting direction inflates Θ_2 and collapses the admissible step. AMOS routes exactly those directions to `OPT1`, where their drift ceases to be charged at all: by Lemma Q.1 the constant Γ_2 is a maximum over S_2 only, and Θ_1 contains no drift constant whatsoever, so a direction's Hessian-Lipschitz parameter disappears from the guarantee entirely once it is reassigned to the first-order optimizer. This is possible because SGD applies no preconditioner and its validity therefore rests on curvature alone.

R LLM USAGE DISCLOSURE

We used LLMs to refine the writing clarity of this paper and for minor assistance with mathematical proof development. All mathematical results and claims have been independently verified by the authors.

S LIMITATIONS AND FUTURE WORK

There is a rich body of theoretical and experimental previous work on the ARC algorithm (Cartis et al., 2012a; 2011a;b; Nesterov & Polyak, 2006; Xu et al., 2020; 2017; Griewank, 1981), but previous works stick to using a single Lipschitz parameter to describe the Hessian drift. It would be interesting to see how integrating our per-eigenspace Lipschitz parameter approach affects that work.

A second direction for future research is a bias correction for our simple Lipschitz parameter estimation in `AMOS`, much like Kingma & Ba (2014) did for Duchi et al. (2011a) and Tieleman & Hinton (2012). Beyond the simple bias correction stemming from zero-initialization of the Lipschitz parameters, `AMOS` also estimates the eigenvalue Lipschitz parameters with the slopes of the eigenvalues computed by the second-order optimizer (instead of the **maximum** of all eigenvalue slopes, as in the definition of Lipschitz continuousness). We do this to reduce computational overhead, however further refinement of this estimation may yield improvements.

A third direction of development is improving `AMOS`'s tensor subdivision strategy. In this work we operated on the tensor granularity, either enabling or disabling the second-order optimizer on each tensor based on its mean Lipschitz parameter. However, it is possible that better results may follow from experimenting with different granularities, e.g. subdividing the tensors into chunks of constant size. No experiments were attempted in this direction.

Fourth, we would like to see future work use appendix C to demonstrate minmax near-optimality of existing and novel optimizers alongside convergence rate proofs. As noted, Wang et al. (2016) provide a simple framework for proofs of convergence, but they fail to demonstrate the usefulness of second-order information; appendix C closes this gap.

Fifth, while the experimental studies in this work focused on a hybridization of SGD and Sophia, a broader experimental study is required to find further well-performing (and perhaps even better-performing) combinations.

Sixth, in this work we have used a rather basic Lipschitz parameter estimation technique in AMOS for computational reasons, approximating Lipschitz parameters with Hessian eigenvalue slopes over Euclidean space (i.e. the denominator is the Euclidean distance between points the Hessian was measured at). More accurate methods of Lipschitz estimation may yield further benefits, however, including modulating the contribution of each dimension to this denominator (it is possible that not all dimensions are equally important to a given eigenspace's Lipschitz estimation denominator) - especially for the ELMO Hessian spectral regularization rule.

One limitation of our Newton's method performance predictor is the additional computational burden of computing the Lipschitz parameters. While some algorithms (e.g. Sophia) explicitly compute all the Hessian eigenvalues, others do not, which poses challenges in computing Lipschitz parameters with minimal overhead. As mentioned, however, since the Hessian is usually only estimated at large intervals, the additional cost of Lipschitz estimation may be amortized across these iterations; furthermore, most existing work uses structures that makes this eigendecomposition more tractable, such as using block-diagonal structure (e.g. Gupta et al. (2018); Jordan et al. (2024); Martens & Grosse (2015); Goldfarb et al. (2020)), which reduces the problem size and exposes opportunities for independent parallel computation.

A second limitation is the need to store a copy of the model parameters at the position at which the previous Hessian was estimated; this adds pressure to the memory, which may pose a bottleneck in settings with large models or activations. Many modern optimizers already hold several buffers the same size as the model, however (e.g. Adam, which stores the first- and second gradient moments), such that this copy does not affect the order of magnitude of storage required. Furthermore, certain training models already store previous copies of the model parameters (e.g. Allen-Zhu (2016)) and we suspect that these may be stored in low precision (since they will only be used to compute the norm of the difference between steps, a simple scalar which is generally robust to low-precision representation), such that this may not add any additional overhead.

A third limitation, in appendix C, is its inability to provide any indication of the number of iterations left to achieve convergence (unlike previous works which bound the gradient norm as a function of the number of iterations, which is a measure of convergence). We see this as an acceptable limitation however, both because there is plenty of existing work aimed at easing convergence proofs and because in practice a model is only required to achieve a certain level of performance on the data decided ahead of time, without regard to how much further it could be optimized. As noted in the introduction, performance is measured by the loss function, so our descent rate bound satisfies this practical requirement.

A fourth limitation of our work is that the performance bottleneck for very large models with large batch sizes may not be the optimization step, but rather the forward- and backward passes, thus limiting AMOS' impact. For smaller models on more modest hardware however, AMOS remains useful. Furthermore, on federated training workloads (training with a very large number of devices, not necessarily localized), the high cost of synchronization (which must be performed at every iteration) motivates usage of sophisticated (and slow) optimizers to minimize the number of iterations; in these cases, AMOS becomes highly relevant. Finally, AMOS helps address the practical overhead barrier that often prevents sophisticated second-order methods from being deployed effectively, potentially bringing such approaches closer to applicability. For instance, Abreu et al. (2025) show convergence in 5.4× fewer iterations than Gauss–Newton approximation baselines when using the full Gauss–Newton inverse, and in 3.4× fewer iterations when using the exact layerwise Gauss–Newton approximation. Yet they also emphasize that the per-iteration computational cost of these methods limits their usefulness in practice. By selectively deploying expensive second-order updates only where they are most worthwhile, AMOS offers a natural path toward making such methods more practical.

We conducted one additional exploratory experiment outside the primary evaluation regime, using GPT-2-XL training on a cluster of 8 A100 GPUs with no tuning or appropriation of AMOS. Standard Adam achieved better performance than standard AMOS in this configuration. However, because this run was not designed as a controlled scaling comparison, we do not interpret it as evidence that Adam generally outperforms AMOS at larger scale, and it is not included among the paper's primary empirical claims. We report it to delimit the scope of our conclusions: the paper makes no claim that AMOS improves over Adam in standard large-model training. A meaningful characterization of the

regimes in which AMOS is preferable would require a factorial study varying model size, batch size, compute budget, and hyperparameter tuning, which is outside the scope of this work.

A fifth limitation is our work's restriction to second-order optimizers. However connections have been shown between certain first- and second-order optimizers, driving interest in applying AMOS to first-order optimizers. For example, while Adam is a first-order method, its mechanistic similarity to Sophia drives interest in viewing its second moments as crude Hessian approximations; likewise, Jordan et al. (2024) discuss the connection between Muon (Jordan et al., 2024) and Shampoo(Gupta et al., 2018).

A sixth limitation is AMOS's applicability to only 2 optimizers. We presented it that way in the paper for simplicity of exposition, but extending it to more optimizers is straightforward - just add more Lipschitz parameter thresholds, and assign a different optimizer to each threshold. This may also be applicable to methods like Hessian-Free (Martens, 2010) which have an adjustable amount of work at each iteration to progressively gather further second-order information; this may be set to be a function of the Lipschitz parameter computed by AMOS for the same effect. Extending it in this way requires retuning the thresholds, however.

A final limitation of our bound is its reliance on $\Delta\Delta^i\theta_t$ as a measure of algorithm optimality which is a function of $\Delta\theta_t^{*\top} v_i$, despite the fact that most optimizers do not compute that during training. We give guidance in appendix C on bridging this gap, but it relies on avoiding the Lipschitz parameter's effect on descent; we hope future work improves this.