IIMB-WP N0. 730/2026

WORKING PAPER NO: 730

Limits of Convergence-Rate Control for Open-Weight Safety

Domenic Rosati
Dalhousie University,
Vector Institute
domenic.rosati@dal.ca

Xijie Zeng
Dalhousie University,
Vector Institute
xijie.zeng@dal.ca

Hong Huang
Dalhousie University,
hn582183@dal.ca

Sebastian Dionicio
Dalhousie University,
sdionicio@dal.ca

Subhabrata Majumdar
Assistant Professor,
Decision Sciences,
Indian Institute of Management Bangalore
smajumdar@iimb.ac.in

Frank Rudzicz
Dalhousie University,
Vector Institute
frank@dal.ca

Hassan Sajjad
Dalhousie University,
hsajjad@dal.ca

Year of Publication – February 2026

Limits of Convergence-Rate Control for Open-Weight Safety

Domenic Rosati [1 2] **Xijie Zeng** [1 2] **Hong Huang** [1] **Sebastian Dionicio** [1] **Subhabrata Majumdar** [3]
Frank Rudzicz [1 2] **Hassan Sajjad** [1]

Abstract

Open-weight foundation models can be fine-tuned for harmful purposes after release, yet no existing training resistance methods provide theoretical guarantees. Treating these interventions as convergence-rate control problems allows us to connect optimization speed to the spectral structure of model weights. We leverage this insight to develop a novel understanding of convergence rate control through spectral reparameterization and derive an algorithm, SpecDef, that can both provably and empirically slow first- and second-order optimization in non-adversarial settings. In adversarial settings, we establish a fundamental limit on a broad class of convergence rate control methods including our own: an attacker with sufficient knowledge can restore fast convergence at a linear increase in model size. In order to overcome this limitation, future works will need to investigate methods that are not equivalent to controlling convergence rate.

1. Introduction

Regardless of how safe foundation models *behave* (**inference-time safety**), there is currently no known method to definitively prevent *training* open weights (Qi et al., 2023) for purposes such as deepfake generation (**training-time safety**). Open-weight governance (Nevo et al., 2024) typically involves pro-open interventions like enhanced user licensing (Seger et al., 2023) and pro-closed interventions like throttled releases (Anderljung et al., 2023). Training-time resistance (Tamirisa et al., 2025; Rosati et al., 2024b; Zheng & Yeh, 2024) offers emerging alternatives, though existing approaches lack theoretical grounding and fail under systematic evaluation (Table 1 and Qi et al., 2024).

Training-time safety can be understood as increasing the number of optimization steps required to fine-tune a model for harmful purposes without degrading its original performance. More formally, we consider an open-weight foundation model $f \in \mathcal{F}$, parameterized by $\theta_{t=0}$. We seek an intervention $\tau \in \mathcal{T} : \mathcal{F} \to \mathcal{F}$ applied ***solely*** at time $t = 0$ that maximizes the number of iterations t required for empirical risk minimization starting with $\tau[f_{\theta_{t=0}}]$ using loss $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ over dataset $\{x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}_i^n$ sampled from a distribution representing unsafe behaviour.

Open-weight **threat models** (Wallace et al., 2025; Casper et al., 2025) consider an attacker with full access to modify pretrained model weights to make it useful for e.g., weapons development. Attackers with budget to train from scratch or distill the model are excluded. In this paper, we distinguish between (1) a non-adversarial **safety regime** where increasing resistance via convergence rate control is desirable for preventing accidental misuse and hardening safety guards (e.g., to enhance responsibility and avoid liability), and (2) an adversarial **security regime** where motivated attackers exist and may have varying levels of skill and budget.

While recent literature (cf. Huang et al., 2024a; Che et al., 2025) proposes a number of intervention methods, they lack a unified theoretical explanation of *why* they provide resistance. This paper provides a foundational theory of training-time safety by framing it as controlling convergence rates in fine-tuning settings. To do this, we utilize classical iteration complexity results for first-order methods (§ 3; second-order extensions in Section J) where curvature of the loss landscape determines loss minimization speed.

Our contributions are:

(i) We develop a tractable method to characterize and control the Hessian spectrum using weight matrices alone (§ 3).
(ii) We identify a class of symmetric transformations that preserve zeroth-order behaviour under arbitrary manipulation of Hessian spectral values (§ 4) and derive an algorithm, SpecDef, that implements these.
(iii) We prove a tight characterization of fundamental limits: a broad class of convergence-rate control methods can be undone at linear cost in model size (§ 6).
(iv) We provide methodological improvements to open-weight safety assessment by evaluating across vision and language domains (Tables 1 and 5), introducing curvature-aware optimizers (Table 7), and deploying spectral repa-

[1]Dalhousie University [2]Vector Institute [3]Indian Institute of Management Bangalore. Correspondence to: Domenic Rosati <domenic.rosati@dal.ca>.

Preprint. February 24, 2026.

rameterization as a novel attack (Section H).

At a high level, our approach is: first-order optimizers must choose learning rates inversely proportional to loss curvature. By inflating weight singular values, we force optimizers into a regime where updates are numerically ineffective.

2. Convergence Rate is Determined By Spectral Values

In this section, we show how to control convergence rates of first-order methods ***even after the release of the model*** by manipulating the principal curvature directions of the optimization objective i.e., the Hessian. It is well known that first-order convergence rates are governed by the Hessian spectrum (Wright & Recht, 2022) (Figure 3 illustrates this).

Consider the sequence of parameter updates $\{\theta_t\}$ from a gradient descent process. The number of iterations $t = k$ required to reduce (i) distance: $\|\theta_t - \theta_*\|$, (ii) loss: $\|\mathcal{L}_{\theta_t} - \mathcal{L}_{\theta_*}\|$, or (iii) gradient: $\|\nabla\mathcal{L}_{\theta_t}\| \leq \epsilon$ is called the *iteration complexity* of an optimization algorithm. In machine learning, θ parameterizes a model f_θ and θ_* is a stationary point of interest such as a ϵ-local minimum or saddle.

2.1. Iteration Complexity Bounds

While stochastic gradient descent and Adam are most commonly used in practice, our analysis focuses on the best-possible first-order iteration complexity given by Nesterov's method of acceleration (Nesterov, 2013). These results assume local convexity and L-smoothness (the first derivative is Lipschitz continuous). For non-convex problems, we can still provide first-order iteration complexity results for L-smooth functions. This assumption is reasonable for foundation model fine-tuning since these networks are often compositions of smooth functions, and nonsmooth results do not improve upon presented rates. In the remainder of this section, we summarize how iteration complexity in both cases depends on the constant L, which is lower bounded by the largest singular value of the Hessian. We omit adaptive, stochastic, and non-smooth methods (see Wright & Recht, 2022; De et al., 2018; Défossez et al., 2022) since they yield worse rates than Nesterov's and still depend on L. Curvature-aware methods such as AdaHessian (Yao et al., 2021), Muon (Jordan et al., 2024), and Sophia (Liu et al., 2023) can achieve better convergence rates, but due to approximation errors and third-order effects, they are still controlled by the methods we develop below (Section J).

Theory Summary: the key quantity governing convergence speed is the smoothness constant L. Larger curvature forces smaller stable learning rates, increasing the number of required iterations. Both worst-case and optimal first-order convergence bounds depend monotonically on L.

Theorem 1 demonstrates sublinear convergence in worst-case smooth settings. We assume that optimal learning rates are chosen; for L-smooth gradient descent, this is $\eta = 1/L$.

Proposition 1 (L-smooth gradient descent iteration complexity)**.** *For gradient descent on an L-smooth loss function $\mathcal{L}$, the number of iterations k needed to achieve $\min_{0\leq i\leq k-1} \|\nabla\mathcal{L}_{\theta_i}\| \leq \epsilon$, is $k \geq (L\|\theta_0 - \theta_*\|)^2/\epsilon^2$, where θ_0 is the initial model parameters, i is the iteration index, and θ_* is the parameters at a stationary point.*

For best-case settings (e.g., starting from θ_0 in a convex basin), Nesterov's optimal rates for first-order methods apply. While it is known that no purely first-order method can have better rates, the complexity is still dependent on L.

Proposition 2 (Nesterov iteration complexity)**.** *Let $\mathcal{L}$ be an L-smooth convex function. For Nesterov's optimal method, the guarantee is that to obtain $\mathcal{L}_{\theta_k} - \mathcal{L}_{\theta_*} \leq \epsilon$, the number of steps is given by $k \geq \sqrt{[2L\|\theta_0 - \theta_*\|^2]/\epsilon} - 1$.*

Both results depend critically on L, which satisfies $L \geq \sigma_1(H_\theta^{\mathcal{L}})$, where $H_\theta^{\mathcal{L}}$ is the loss Hessian (shown in Theorem 13). Detailed proofs, and reviews of L-smooth gradient descent and Nesterov's method, appear in Section A. Analysis of these rates reveals three opportunities for slowing down the iteration complexity in the following section.

2.2. Controlling Convergence Rates

Bad local minima The first opportunity to control convergence rates is to seek high-loss local minima from which gradient descent cannot escape. However, random matrix theory for deep networks shows that such bad minima are rare and may not always exist (Choromanska et al., 2015; Baskerville et al., 2022b;a). Even when they exist, finding them is computationally intractable (Bubeck & Mikulincer, 2020; Hollender & Zampetakis, 2023; Huanjian et al., 2025). Furthermore, methods like stochastic gradient Langevin dynamics can escape such local minima (Raginsky et al., 2017; Tzen et al., 2018), making this approach easy to defeat.

Weight distance The second approach is to maximize the weight distance $\|\theta_0 - \theta_*\|$. However, in foundation model fine-tuning, the pre-trained weights are already close to task-specific optima (Neyshabur et al., 2020), making this distance inherently small. Gradient ascent methods explicitly designed to maximize this distance have proven ineffective (Rosati et al., 2024b; Golatkar et al., 2019).

Increasing curvature The final option is to manipulate L. Increasing L forces smaller learning rates for convergence. We show later that by making L sufficiently large, learning rates can be driven into a regime where gradient updates become numerically ineffective, resisting convergence.

Beyond these approaches, one might consider obfuscating gradients with high variance to prevent unbiased estimation. However, this has been shown to be easily defeated in Athalye et al. (2018). Conversely, naively reducing curvature to create flat landscapes is unlikely to succeed given that it would allow larger stable learning rates. Although analysis of previous methods in Section H reveals that a more nuanced curvature reduction based on singular vector alignment might provide some training time resistance.

3. A Hessian Spectral Lower Bound Dependent on Weight Spectrum

We now state our main theoretical result. The Hessian is denoted $H_\theta^{\mathcal{L}} \equiv \nabla_\theta^2 \mathcal{L} = \left[\frac{\partial^2 \mathcal{L}}{\partial \theta_i \partial \theta_j}\right]_{i,j}$ where, in this context, θ is the concatenated vector of all parameters and i, j index Hessian blocks or the concatenated vector of a single layer (θ_i). The Hessian, computed for a neural network function $f_\theta(x) : x \mapsto \theta_{n+1} \circ \phi_n \circ \theta_n \circ \cdots \circ \phi_1 \circ \theta_1 x$ with element-wise activations $\phi : \mathbb{R}^d \to \mathbb{R}^d$, admits a lower bound on its largest singular value depending on $\sigma_1(\theta_i)$ of a chosen weight matrix. Controlling $\sigma_1(\theta_i)$ thus controls $\sigma_1(H_\theta^{\mathcal{L}})$ and, via Theorem 13, the convergence rate. An illustrated intuition of the principal angle quantity $\cos\theta(A, B)$ (see Theorem 18) that we use below can be found in Figure 4.

Theorem 3 (Hessian Singular Value Lower Bound)**.** *Let $\nabla_\theta^2 \mathcal{L} \in \mathbb{R}^{n \times n}$ be the Hessian of $\mathcal{L}$ with network function defined earlier. Assume there exists a submatrix of the Hessian $\nabla^2_{\theta_i,\theta_j} \mathcal{L} := \left[\frac{\partial^2 \mathcal{L}}{\partial \theta_i \partial \theta_j}\right]_{i,j} \in \mathbb{R}^{p \times q}$ where $p = |\theta_i|, q = |\theta_j|$ that has the matrix product structure ABC with arbitrary matrices A, C. Then there exists a $B := \theta_k$ such that $\sigma_1(\nabla_\theta^2 \mathcal{L})$ is bounded below as*

$$\sigma_1(\nabla_\theta^2 \mathcal{L}) \geq \sup_{r_1, r_2 \geq 1} \sigma_{r_1}(A)\sigma_1(B)\sigma_{r_2}(C) \cos\theta_1 \cos\theta_2,$$

where $\cos\theta(\cdot,\cdot)$ is the cosine of the largest principal angle between the subspace as defined in Theorem 18 with $\cos\theta_1 := \cos\theta(A_{r_1}, BC_1)$ and $\cos\theta_2 := \cos\theta(B_1, C_{r_2})$.

Proof Sketch. The full proof appears in Section D.2; see Section B for background on singular value inequalities. Hessian spectral values are bounded below by spectral values of submatrices (via Poincaré's separation theorem, Theorem 23). These submatrices correspond to second partial derivatives w.r.t. parameter matrices θ_i and θ_j. These block matrices decompose following the algebraic structure above, yielding the lower bound via Theorem 19. □

Standard architectures (MLPs, CNNs, Transformers) satisfy similar bounds (Section D.2). The tightness of the bound depends on the alignment of the matrix products that compose the Hessian and is extensively discussed in Theorem 25 and Section C.1 with the implication that small principal angles resulting from rank deficiency or alignment may require selecting more layers or additional singular value indices. This bound is tighter than classical matrix analysis results (Horn & Johnson, 1991; Bhatia, 2013) and can uniquely provide non-vacuous results under rank deficiency.

Example 4 (Three-layer MLP). For a three-layer network with ReLU activations, the Hessian block w.r.t. θ_3, θ_1 is given by

$$\frac{\partial^2 f}{\partial \theta_3 \partial \theta_1} = \underbrace{(x^\top \otimes I_m)^\top D_{z_1}}_{A} \underbrace{\theta_2^\top}_{B} \underbrace{D_{z_2}}_{C}.$$

Here, A and C are activation (z_1, z_2) dependent Jacobian (D) factors, while $B = \theta_2^\top$ is the intermediate weight matrix. Consequently, the spectrum of this Hessian block is governed by the singular values of θ_2, together with the spectra of the Jacobian factors and the principal angles.

We provide two empirical validations of the bound. First, Table 11 illustrates Theorem 3 using Hessian derivations from Section D.1. Second, Figure 1 shows how $\sigma_1(H_\theta^{\mathcal{L}})$ and convergence rate evolve as $\sigma_1(\theta_i)$ increases in practical architectures. These results confirm that controlling weight matrix spectral values yields convergence rate control. Details on the numerical analysis appear in Section F.

4. Spectral Reparameterization for Convergence Control

Having established that convergence rates depend on weight matrix singular values, we now ask: how can we manipulate them algorithmically? We first formalize the class of necessary transformations and derive from it SpecDef (Example Construction 1). Arbitrary spectral modifications can destroy model behaviour, which is unacceptable in foundation models. We therefore seek a transformation $\mathcal{T}$ that can: (1) increase the largest Hessian singular value σ_1 arbitrarily, and (2) keeps zeroth-order behaviour (e.g., the function output or loss values) unchanged. Such a transformation is a second-order spectral symmetry, preserving zeroth-order behaviour under Hessian spectrum deformations. Formally, let $f(x) = \theta_{n+1} \circ \phi_n \circ \theta_n \circ \cdots \circ \phi_1 \circ \theta_1 x$ be a composition of linear transformations $s_i = \theta_i z_{i-1}$ with weight matrices $\theta_i \in \mathbb{R}^{d_i \times d_{i-1}}$ and twice-differentiable activation functions $\phi_i : s_i \mapsto z_i$, where i is the layer index and $z_0 = x \in \mathbb{R}^d$. The spectral values of θ_i appear in $\Sigma_i = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\min\{d_i, d_{i-1}\}})$ from the SVD $\theta_i = U_i \Sigma_i V_i^\top$ with $\theta_i \in \mathbb{R}^{d_i \times d_{i-1}}$.

Definition 5 (Lower-Max Spectral Reparameterization)**.** For a given function composition f as defined above, a constant $c > 0$ which will define our minimum spectral value, and $\epsilon \geq 0$ which will define our maximal function distance. A map $\mathcal{T}_c : f_\theta \mapsto f_{\theta'}$ is a lower-max spectral

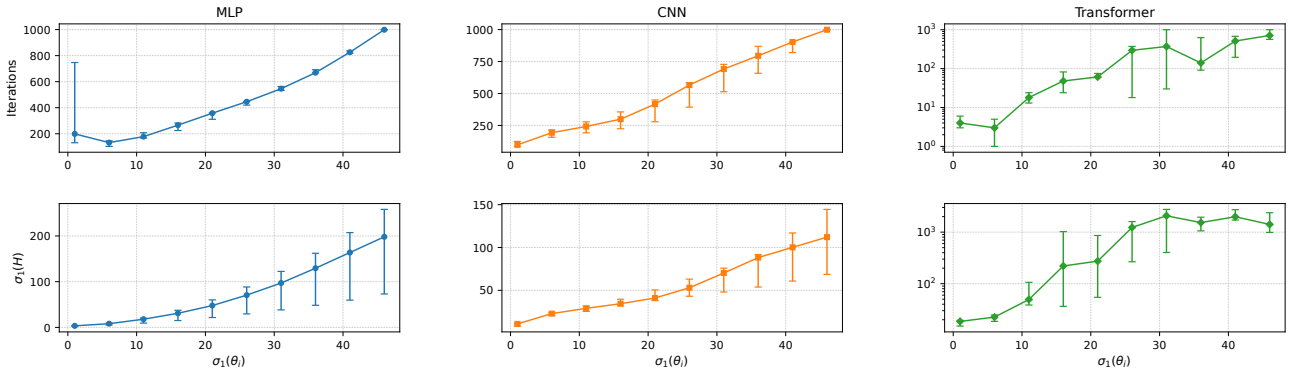


Figure 1. Convergence Rate Control: $\sigma_1(H_\theta^{\mathcal{L}})$ and convergence rate (iterations) is increased with $\sigma_1(\theta_i)$

reparameterization of f if

1. **Spectral Control** $\sigma_1(H_{\theta'}^{\mathcal{L}}) \geq c$ where $H_{\theta'}^{\mathcal{L}}$ is the Hessian w.r.t. the parameters of $\mathcal{T}_c[f_\theta]$.
2. **Functional Invariance up to** ϵ Given a distance function $d(f, g)$ on the space of functions $f, g \in \mathcal{F}$, $d(\mathcal{T}_c[f], f) \leq \epsilon$ for a small $\epsilon \geq 0$.

This definition captures transformations that leave model predictions unchanged while making optimization arbitrarily slow. We consider only a lower bound on the maximum Hessian spectral value—a more general version with upper bounds appears in Theorem 33, used later for attack construction. Importantly, in this paper we only consider **global convergence rate control for all distributions**, though future definition could be relaxed to specific distributions over $\mathcal{L}$.

Our observations build on prior work in symmetric reparameterizations, sharpness, and generalization (Dinh et al., 2017; Foret et al., 2020). In contrast to scale invariance analyses, control here arises from a small top-k singular value, indicating a deeper symmetry. Kristiadi et al. (2023) studied related geometric effects of reparameterization on first- and second-order properties, but applied them to improving optimization. Symmetry-based perspectives are now standard in optimization for machine learning (Simsek et al., 2021; Zhao et al., 2022), which we apply to open-weight safety.

Example Construction 1 presents Spectral Deformation (SpecDef) as a spectral reparameterization that can directly be obtained from Theorem 6 by first inserting identity linear layers adjacent to θ_j, then applying a compensation update. Intuitively, SpecDef works by scaling selected singular values and inserting compensating layers so that the overall function remains unchanged. The algorithm runs once before model release, and is fast: SVD applies only to a few selected layers, which can be batched on GPU, and the matrices are small. For `GPT-OSS-20b` with 10 layers, this takes under 15 seconds. Further orthogonal decompositions like QR may be used to improve speed of numerical stability.

Example Construction 1 Spectral Deformation (SpecDef)

Model parameters $M = \{\theta_i \in \Theta\}$ where i is a layer index, $n \in \mathbb{N}$ number of layers to select, k is the top k singular values to select, α is the singular value multiplier to use, and layer selection function Select : $\Theta \times \mathbb{N} \to \Theta$.
$\Theta' \leftarrow \text{Select}(\Theta, n)$ *; e.g., random selection*
for $\theta_i \in \Theta'$ **do**
  $M \leftarrow [M_{i+1:}; I; M_{:i}]$ *; Inject identity layer*
  $U, \Sigma, V \leftarrow \text{SVD}(\theta_i)$
  $T \leftarrow \text{Diag}(\alpha, \ldots, \alpha, 1, \ldots, 1)$ *; Indices* $1..k$ *get* α
  $\widetilde{\Sigma} \leftarrow T\Sigma$
  $\theta_i' \leftarrow U\widetilde{\Sigma}V^\top$
  $\theta_i^{comp} \leftarrow U\Sigma\widetilde{\Sigma}^{-1}U^\top$ *; or pseudoinverse*
  $M \leftarrow [M_{i+1:}; \theta_i^{comp}; \theta_i'; M_{:i-1}]$ *; replace* I, θ_i
end for

Theorem 6 (SpecDef is a (Lower-Max) Spectral Reparameterization)**.** *The following spectral deformation is a (lower-max) spectral reparameterization: For θ_i, set $\widetilde{\Sigma}_{\theta_i} \leftarrow T\Sigma_{\theta_i}$ such that $\sigma_1(U\widetilde{\Sigma}_{\theta_i}V^\top) = \alpha$. Where α is set so that $\sigma_1(H_\theta^{\mathcal{L}}) \geq c$ (Spectral Control) where $c > 0$ is given. Next, compensate either with $\theta_{j>i} \leftarrow \theta_j U_{\theta_i}\Sigma_{\theta_i}\widetilde{\Sigma}_{\theta_i}^{-1}U_{\theta_i}^\top$ or $\theta_{j<i} \leftarrow V_{\theta_i}\widetilde{\Sigma}_{\theta_i}^{-1}\Sigma_{\theta_i}V_{\theta_i}^\top\theta_j$ (Functional Invariance). Where $\phi_{i:j}$ and $\phi_{j:i}$ must be degree-1 homogeneous and $\widetilde{\Sigma}^{-1}$ is the pseudoinverse in the case of rectangular matrices.*

The proof appears in Section E and follows simply from the application of $\sigma_1(U\widetilde{\Sigma}_{\theta_i}V^\top) = \alpha$ implying $\sigma_1(H_\theta^{\mathcal{L}}) \geq c$ by Theorem 3. Standard architectures typically are not composed of purely 1-homogeneous activation functions preventing cross-layer compensation (at least without adaptation which can be obtained ala Mirzadeh et al., 2023). For such architectures, we use layer injection: inserting an adjacent layer with the identity activation, a special case of Theorem 6.

Table 1. Relearning attack evaluation. Each entry considers number of training steps and WMDP-bio (Li et al., 2024) final accuracy in brackets. Dagger (†) indicates perplexity increased by more than 100%. All methods use `Llama-3.1-8B-Instruct`. SpecDef is applied to the `ELM`-unlearned model. The unfiltered base model for `DeepIgnorance` has an effective LR floor starting at 10^{-5}.

	START ACC	10^{-6}	5×10^{-6}	8×10^{-6}	10^{-5}	3×10^{-5}
ELM	0.204	120 (0.607 ± 0.008)	33 ± 5 (0.635 ± 0.004)	30 (0.634 ± 0.013)	23 ± 5 (0.621 ± 0.027)	20 (0.671 ± 0.014)
REPNOISE	0.325	510 (0.468 ± 0.044)	40 (0.629 ± 0.005)	30 (0.616 ± 0.007)	30 (0.627 ± 0.008)	33 ± 15 (0.650 ± 0.023)
RMU	0.259	510 (0.558 ± 0.022)	40 (0.625 ± 0.009)	33 ± 5 (0.626 ± 0.018)	30 (0.634 ± 0.022)	23 ± 5 (0.651 ± 0.030)
RR	0.251	300 ± 181 (0.595 ± 0.009)	33 ± 5 (0.609 ± 0.013)	30 (0.639 ± 0.020)	23 ± 5 (0.610 ± 0.010)	23 ± 5 (0.643 ± 0.008)
TAR	0.290	510 (0.307 ± 0.008)	126 ± 20 (0.609 ± 0.006)	76 ± 5 (0.620 ± 0.014)	66 ± 5 (0.638 ± 0.035)	66 ± 25 (0.618 ± 0.010)
GRADDIFF-WMDP	0.227	510 (0.444 ± 0.030)	223 ± 248 (0.574 ± 0.067)	90 ± 60 (0.618 ± 0.022)	70 ± 26 (0.616 ± 0.017)	50 ± 17 (0.624 ± 0.020)
NPO-SAM	0.275	510 (0.456 ± 0.014)	73 ± 11 (0.605 ± 0.006)	46 ± 5 (0.617 ± 0.009)	43 ± 5 (0.626 ± 0.008)	63 ± 11 (0.651 ± 0.020)
NPO	0.282	510 (0.506 ± 0.022)	66 ± 11 (0.613 ± 0.010)	46 ± 11 (0.612 ± 0.008)	40 (0.616 ± 0.007)	46 ± 25 (0.631 ± 0.024)
SIMNPO	0.337	510 (0.499 ± 0.022)	63 ± 5 (0.607 ± 0.006)	43 ± 5 (0.629 ± 0.005)	40 ± 10 (0.629 ± 0.029)	30 ± 10 (0.618 ± 0.010)
DEEPIGNORANCE (STRONG FILTER)	0.408	–	–	–	510 (0.476 ± 0.013)	126 ± 20 (0.618 ± 0.016)
LLAMA-3.1-8B	0.482	93 ± 5 (0.607 ± 0.006)	30 (0.651 ± 0.010)	20 (0.618 ± 0.019)	20 (0.629 ± 0.005)	10 (0.609 ± 0.005)
SPECDEF $\alpha = 10$k	0.204	10 (0.195 ± 0.065)‡	10 (0.171 ± 0.039)‡	10 (0.176 ± 0.020)‡	10 (0.191 ± 0.061)‡	10 (0.195 ± 0.024)‡

Implementation Details To connect SpecDef to convergence control, the smallest effective learning rate, i.e., the rate below which gradients become ineffective, can be determined empirically (Section G). Operationally, we are forcing the attacker to use this rate, since higher rates would cause divergence. The multiplier α to increase σ_k for the top k singular values of selected layers can then start at the reciprocal of this rate. For example, many language models fail to learn below $\eta = 10^{-6}$ using SGD or Adam, so we can start with 10^6, or higher to account for the other factors in the bound in Theorem 3. Layer selection and how much to increase the singular value multiplier can be determined empirically with test losses and hyperparameter sweeps as part of a "certification" process. In our experiments, we find several cases where the singular value can be much lower than the reciprocal value of the effective learning rate, and only a handful of layers are needed. Setting k to be higher gives more opportunities to align the tail of σ_k with the largest singular values in the matrix product (Theorem 25). Finally, readers may remark that SpecDef is not secure in that an adversary may combine the compensated layers. We don't attempt a spectral reparameterization with *stealthy* properties (though this may be obtained) due to the results of (**?**) which show a general attack on spectral reparameterization methods.

5. Experimental Validation

We now validate SpecDef (and, by extension, our theoretical framework) across four attack scenarios: relearning attacks on unlearned models, harmful fine-tuning, NSFW content generation, and artistic style transfer. Experiments span language models (`Llama-3.1-8B`, `GPT-OSS-20B`) and vision models (`Stable Diffusion V1-4`). We compare against 10 prior defences for relearning and the strongest known methods for each other domain (`TAR`, `RepNoise`, `IMMA`). We also evaluate curvature-aware optimizers (Sophia, Muon, AdaHessian) as adaptive attacks.

The key findings are: (1) all prior methods fail under standard fine-tuning when learning rates are varied (Table 1), (2) SpecDef induces divergence across all tested learning rates at sufficient α (Tables 4 to 6 and Figure 2), (3) curvature-aware optimizers do not circumvent resistance (Table 7), and (4) model utility is preserved (Tables 2 and 3).

Table 2. SpecDef preserves model utility. Performance differences are negligible across benchmarks even at extreme multipliers.

α	WMDP	PPL	MMLU	WINOGRANDE	ARC	HELLASWAG
1K	Δ=-0.1	Δ=-0.01	Δ=+0.1	Δ=+0.0	Δ=+1.0	Δ=+0.3
100K	Δ=-0.1	Δ=-0.01	Δ=+0.6	Δ=+0.3	Δ=+0.3	Δ=+0.0
1B	Δ=+0.0	Δ=-0.02	Δ=+0.1	Δ=-1.0	Δ=+1.0	Δ=+0.0

Setup All experiments use AdamW with standard `PyTorch` hyperparameters unless noted. The multiplier α scales the top-k=25 singular values of 5 randomly selected layers; α=1 is unmodified. We report results across 3 random seeds. We sweep learning rates and mark divergence with † (at least one seed) or ‡ (all seeds). Divergence indicates that perplexity has doubled. As per Henderson et al. (2023), model collapse counts as successful. The tables should be read as follows: each cell reports the mean steps it takes to reach a task score of $> 60\%$ or to diverge. The value in brackets is the mean score on the final step. Full experimental details are reported in Section F.

Table 3. Style relearning: SpecDef prevents Van Gogh style recovery; ESD and IMMA fail within 10 epochs of LoRA.

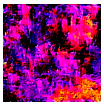
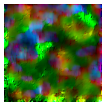
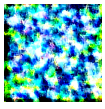
METHOD	BEFORE	1 EPOCH	10	50
ESD				
IMMA				
SPECDEF				

Table 4. Relearning and harmful fine-tuning attacks on `Llama-3.1-8B` models. SpecDef causes divergence (‡) at high enough α.

Unlearning with WMDP-Bio (elm-unlearned Llama-3.1-8B)							
Learning Rate	$\alpha = 1$	1k	2500	5k	7500	10k	1M
10^{-6}	120 ± 10 (0.616 ± 0.010)	176 ± 288 (0.424 ± 0.098)†	46 ± 56 (0.305 ± 0.051)‡	16 ± 10 (0.231 ± 0.077)‡	16 ± 10 (0.222 ± 0.078)‡	13 ± 5 (0.204 ± 0.068)‡	10 (0.216 ± 0.062)‡
5×10^{-6}	33 ± 5 (0.644 ± 0.054)	26 ± 28 (0.278 ± 0.168)‡	13 ± 5 (0.243 ± 0.095)‡	10 (0.197 ± 0.055)‡	10 (0.187 ± 0.032)‡	10 (0.224 ± 0.073)‡	10 (0.182 ± 0.047)‡
8×10^{-6}	26 ± 5 (0.637 ± 0.022)	20 ± 17 (0.267 ± 0.164)‡	13 ± 5 (0.222 ± 0.101)‡	10 (0.194 ± 0.061)‡	10 (0.303 ± 0.125)‡	10 (0.216 ± 0.109)‡	10 (0.179 ± 0.036)‡
10^{-5}	26 ± 5 (0.658 ± 0.030)	16 ± 11 (0.243 ± 0.146)‡	10 (0.221 ± 0.087)‡	10 (0.200 ± 0.033)‡	10 (0.195 ± 0.026)‡	10 (0.217 ± 0.039)‡	10 (0.201 ± 0.069)‡
3×10^{-5}	33 ± 23 (0.651 ± 0.027)	10 (0.218 ± 0.079)‡	10 (0.174 ± 0.033)‡	10 (0.183 ± 0.059)‡	10 (0.217 ± 0.059)‡	10 (0.190 ± 0.016)‡	10 (0.184 ± 0.030)‡
Harmful Fine-tuning with Beavertails (Llama-3.1-8B)							
Learning Rate	$\alpha = 1$	1k	2500	5k	7500	10k	1M
10^{-6}	293 ± 11 (0.619 ± 0.005)	33 ± 15 (0.674 ± 0.044)†	20 ± 10 (0.650 ± 0.042)†	13 ± 5 (0.627 ± 0.130)†	13 ± 5 (0.616 ± 0.126)†	13 ± 5 (0.617 ± 0.127)†	10 (0.043 ± 0.040)‡
5×10^{-6}	96 ± 5 (0.689 ± 0.069)	13 ± 5 (0.602 ± 0.144)†	13 ± 5 (0.519 ± 0.274)†	10 (0.275 ± 0.369)†	10 (0.267 ± 0.418)†	10 (0.284 ± 0.431)†	10 (0.023 ± 0.019)‡
8×10^{-6}	76 ± 5 (0.701 ± 0.015)	13 ± 5 (0.606 ± 0.152)†	10 (0.214 ± 0.347)†	10 (0.275 ± 0.414)†	10 (0.266 ± 0.443)†	10 (0.278 ± 0.429)†	10 (0.037 ± 0.060)‡
10^{-5}	63 ± 5 (0.651 ± 0.031)	13 ± 5 (0.622 ± 0.138)†	10 (0.257 ± 0.368)†	10 (0.264 ± 0.436)†	10 (0.282 ± 0.432)†	10 (0.268 ± 0.441)†	10 (0.029 ± 0.047)‡
3×10^{-5}	40 (0.747 ± 0.041)	10 (0.249 ± 0.400)†	10 (0.262 ± 0.449)†	10 (0.360 ± 0.392)†	10 (0.238 ± 0.333)‡	10 (0.159 ± 0.211)‡	10 (0.028 ± 0.039)‡

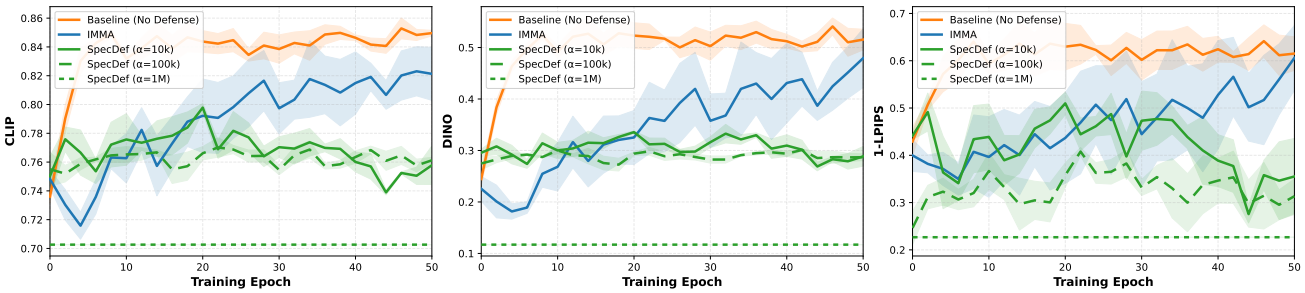


Figure 2. Style transfer attack: SpecDef maintains low similarity to Van Gogh references throughout 50 epochs of LoRA fine-tuning, while ESD and IMMA recover the erased style within 10 epochs. Lower similarity indicates stronger resistance.

Table 5. NSFW generation: SpecDef achieves 100% reduction in nudity counts; IMMA achieves 63%.

Method	Nudity Counts ↓	Images w/ Nudity ↓	Img. Reduction ↑	Nud. Reduction ↑
Vanilla SD V1-4	346.0	205 / 205	—	—
Nudity-Erased SD (ESD)	59.0	32 / 205	84.4%	82.9%
Unprotected ESD w/ LoRA	311.0 ± 23.4	119.3 ± 2.9 / 205	41.8 ± 1.4%	10.1 ± 6.8%
IMMA (Zheng & Yeh, 2024)	127.7 ± 42.7	62.3 ± 14.1 / 205	69.6 ± 6.9%	63.1 ± 12.4%
SpecDef $\alpha = 10$k	**0.0 ± 0.0**	**0.0 ± 0.0 / 205**	**100.0 ± 0.0%**	**100.0 ± 0.0%**

Table 6. Scaling to 20B parameters with LoRA: training resistance is possible on large models with high enough α.

	LR = 10^{-4}	LR = 5×10^{-4}	LR = 10^{-3}
Baseline	580 (0.61 ± 0.00)	236 ± 37 (0.62 ± 0.02)	193 ± 9 (0.62 ± 0.02)
$\alpha = 1000$	16 ± 4 (0.67 ± 0.04)†	10 (0.57 ± 0.23)†	10 (0.26 ± 0.23)‡
$\alpha = 10,000$	10 (0.26 ± 0.27)‡	10 (0.00 ± 0.00)‡	10 (0.00 ± 0.00)‡

Comparison with baselines Unlearning robustness against relearning attacks is the most well-developed setting in training-time safety (Che et al., 2025; Fan et al., 2025; O'Brien et al., 2025) so it is our primary focus for the analysis of baseline methods. Table 1 shows no previous methods are able to withstand standard fine-tuning as learning rates are varied. While varying starting accuracies could theoretically confound convergence comparisons, all baseline methods eventually reach the target accuracy regardless of starting point and SpecDef does not give sufficient α.

Our results also hold in the case for harmful fine-tuning with `TAR` (Tamirisa et al., 2025) and `RepNoise` (Rosati et al., 2024a) (Section F.2.2) as well as `IMMA` (Zheng & Yeh, 2024) for NSFW training (Table 5) and style transfer attacks (Figure 2). For SpecDef, we present comprehensive sweeps of learning rates of our method for unlearning and harmful fine-tuning in Table 4 and vision experiments in Table 21 showing that as long a sufficient α multiplier is used then convergence rate control is achieved. Table 6 corroborates our findings in a large 20b parameter setting using LoRA which reveals a phase transition: at $\alpha = 1$k with 5 layers, the defence fails to prevent convergence, while $\alpha = 10$k consistently induces divergence. This identifies the minimum effective α for multi-layer configurations. A more comprehensive study of past methods and where they fit within our framework as well as a novel attack that defeats them based our framework is presented in Section H.

For adaptive optimizers, convergence rate can be initially accelerated as we increase α. This is expected because the gradient can now be much larger and if the step sizes are not diverging then faster convergence would occur. Small multipliers are not intended to provide resistance as our method is intended to operate in the high-α regime where divergence occurs. Section H shows cases where this acceleration can be leveraged to attack previous defences.

Curvature-aware methods We evaluate three classes of curvature-aware optimizers: Sophia (Liu et al., 2023) (second-order approximate method with Hessian diagonal estimation using Gauss-Newton-Bartlett, $\beta_1 = 0.965$, $\beta_2 = 0.99$, $\rho = 0.04$), Muon (Jordan et al., 2024) (geometry-aware method using Newton-Schulz orthogonalization with momentum 0.95), and AdaHessian (Yao et al., 2021) (second-order approximate method using Hutchinson's diagonal Hessian estimator). Table 7 presents the

Table 7. Curvature-aware optimizers fail to circumvent SpecDef. Muon performs best but still diverges at $\alpha \geq 100$k; Sophia is ineffective even at $\alpha = 1$k.

	MUON		
α	10^{-5}	2×10^{-5}	5×10^{-5}
1	765 (0.29 ± 0.05)	765 (0.43 ± 0.03)	316 ± 45 (0.61 ± 0.01)
1K	335 ± 375 (0.53 ± 0.15)	305 ± 399 (0.54 ± 0.11)	140 ± 156 (0.62 ± 0.03)
10K	308 ± 396 (0.50 ± 0.13)†	83 ± 66 (0.48 ± 0.11)‡	36 ± 11 (0.41 ± 0.09)‡
100K	23 ± 5 (0.40 ± 0.10)‡	20 ± 10 (0.37 ± 0.10)‡	13 ± 5 (0.33 ± 0.11)‡
	SOPHIA		
α	5×10^{-6}	10^{-5}	2×10^{-5}
1	36 ± 5 (0.63 ± 0.02)	60 ± 60 (0.63 ± 0.02)	36 ± 28 (0.58 ± 0.13)†
1K	13 ± 5 (0.25 ± 0.13)‡	13 ± 5 (0.21 ± 0.07)‡	10 (0.19 ± 0.05)‡
	ADAHESSIAN		
α	10^{-6}	3×10^{-6}	5×10^{-6}
1	126 ± 11 (0.62 ± 0.01)	60 (0.64 ± 0.00)	66 ± 15 (0.66 ± 0.04)
1K	153 ± 230 (0.47 ± 0.17)†	30 ± 34 (0.31 ± 0.08)‡	20 ± 17 (0.22 ± 0.05)‡
10K	13 ± 5 (0.19 ± 0.03)‡	10 (0.23 ± 0.05)‡	10 (0.22 ± 0.03)‡

results of these methods. The same setting as Table 4 is used. Our findings were: (i) Sophia was largely ineffective at undoing SpecDef, (ii) AdaHessian was more effective than Sophia but less effective than Muon, and (iii) Muon's ability to undo SpecDef was roughly comparable to the earlier Adam-based attacks with some cases where increasing α below the subcritical regime can accelerate Muon. Section J provides a thorough empirical and theoretical analysis of why this is the case. Unlike SpecDef, previous training resistance methods were very easy to undo with curvature-aware methods (see Table 25) and therefore these should be to strengthen methodology in future studies.

Ablation experiments on layer selection and top-k selection are presented in Section G. We tried several natural adaptive attacks such as quantization and naive rescaling but did not report them as they immediately destroy numerical stability due to the poor conditioning SpecDef introduces.

6. Fundamental Limits of Convergence-Rate-Based Resistance

While convergence-rate control imposes practical friction, it provides only limited security guarantees. To this end, we prove a fundamental limitation: any mechanism that slows fine-tuning by changing convergence rates can be efficiently bypassed by an adaptive attacker. In particular, methods that control convergence via weight matrix singular values—the only viable approach for reparameterization under standard assumptions (e.g., Lipschitz network components; see Theorem 37)—can be defeated given sufficient attacker budget and expertise. Such methods exhibit a fundamental trade-off: an attacker can achieve a k-th root reduction in $\max_i \sigma_1(\theta_i)$ by training a model k times larger.

To this end, we first establish universality.

Theorem 7 (Only Weight Matrices Provide Unbounded Spectral Control)**.** *Consider a feed forward network f as defined in §4. Additionally assume the activation functions ϕ to be approximately non-expansive, with approximately non-expansive first two derivatives if they exist. Also, let the data be bounded with $\|x\|_2 \leq c$ for any x drawn from the dataset $\mathcal{D}$. A transformation of that function $\mathcal{T} : \mathcal{F} \to \mathcal{F}$ is a lower-max spectral reparameterization (i.e., a $(l, \infty, 1)$-spectral reparameterization) only if the transformation is a reparameterization of the parameter matrices θ_i.*

Proof Sketch. The proof (Section I.2) proceeds from two standard assumptions (see Du et al., 2019; Xiong et al., 2020). (1) bounded data: token embeddings are bounded by construction, pixel normalization ensures boundedness, training datasets have finite maximum norms, (2) nonlinear components have bounded Jacobian and Hessian norms. This includes element-wise activations, residual connections, and normalization layers under standard variance-floor assumptions; attention mechanisms are Lipschitz with constants depending multiplicatively on query, key, and value weight norms. Under these conditions, we show by contradiction that Hessian spectral values must be controlled by weight matrix factors. □

Having established this universal class of convergence-rate control methods, we now show that all such methods can be undone by an adversary—a fundamental limitation of possible security guarantees.

Theorem 8 (Cost of Undoing Spectral Reparameterization)**.** *Assume the feed forward networks defined in §4. Let $M = \max_{1 \leq i \leq n} \sigma_1(\theta_i)$. Then to achieve $\max_i \sigma_1(\theta_i) \leq M^{1/k}$ it suffices to insert at most $(k-1) \cdot n$ additional linear layers and perform spectral deformation.*

Proof Sketch. The proof and practical details are given in Section I. Assuming that the defender controls the convergence rate by maximizing $\{\sigma_1(\theta_i)\}_i$ (for which we do not know alternatives), we show an upper bound analogous to Theorem 3. An attacker achieves k-th root reduction by factoring each layer into k layers using Theorem 6. □

Extensions to common deep architectures follow under mild technical assumptions (Theorem 37). While worst-case attacks can be made expensive, these results show that lower-max spectral reparameterizations have limited security guarantees. By Theorem 7 and Section 2.2, any open-weight safety method maximizing convergence time for unsafe objectives is subject to this attack. Cheaper attacks are unlikely given that spectral control directly bounds iteration complexity. For example (1) naively scaling layers down might give more tractable gradients but now σ_n causes conditioning that is very poor which would result in instability and (2) naive quantization or clipping would not capture the necessary numerical range of the original transformation

effectively destroying model utility. Upper-max spectral reparameterizations that flatten curvature are even easier to defeat (we demonstrate this in Section H).

Experiments We now empirically evaluate the minimal *attack budget* required to undo convergence-rate control. The attacker restores original training cost via Theorem 8 by inserting $k-1 \cdot n$ identity layers and factorizing defended layers via SVD to achieve $\sigma_1^{1/k}$. We first evaluate relearning attacks on WMDP-Bio for unlearning with learning rate fixed at 3×10^{-5}. The α multiplier varies across columns; attack cost (number of added layers, resulting in linear parameter increase) varies across rows.

As predicted, Table 8 shows that required compute grows linearly in k and logarithmically in the defender's singular-value multiplier, with WMDP-Bio accuracy recovering within few steps after (but not before) appropriate parameter addition. Due to compute limitations, for `Llama-3.1-8B` with `ELM` unlearning we only demonstrate spectral deformation on one layer.

We now empirically examine the attack on a fully reparameterized model. We evaluate the harmful fine-tuning attack on `SmolLM2-360M-Instruct`, which includes a safety guard where the model does not generate tokens when the query is harmful. In the absence of reparameterization, this safety guard can be undone within 10 fine-tuning steps, achieving over 70% harmfulness.

When we reparameterize all layers with $\alpha = 100$k using the smallest effective learning rate (10^{-7}), undoing SpecDef requires increasing the model size by approximately $4\times$. These model size increases are relative to post-SpecDef application where SpecDef initially doubles the model size with injected layers (360m to 876m). For $\alpha = 1$M, undoing SpecDef requires approximately $6\times$ as many parameters and $10\times$ for $\alpha = 10$M, increasing VRAM usage from 1.63GB to 9.30GB with 4.99B total parameters (some layers like LM head are ignored). This translates into an increase from 2.7 average seconds per step to 7.5 seconds per step. In Table 8, fewer injected layers are required to undo these α multipliers. This occurs because when only a single layer (or a few layers) is reparameterized, the network can tolerate one or two, moderate but higher than the original, singular values without destabilizing the overall dynamics. However, when many layers are reparameterized, even moderate singular values across multiple layers compound multiplicatively, leading to large activations and gradient norms.

Table 8. Layer injection attack validates Theorem 8: adding $k-1$ layers reduces effective resistance from α to $\alpha^{1/k}$, restoring convergence at linear cost in parameters.

LAYERS ADDED	$\alpha = 1$	$\alpha = 10^5$	$\alpha = 10^6$	$\alpha = 10^9$
0 ($\sigma^{1/k=1}$)	80 (0.61)	10 (0.16)†	10 (0.15)†	10 (0.18)†
1 ($\sigma^{1/k=2}$)	80 (0.62)	130 (0.62)	10 (0.21)†	10 (0.17)†
2 ($\sigma^{1/k=3}$)	30 (0.64)	80 (0.61)	100 (0.60)	10 (0.17)†
3 ($\sigma^{1/k=4}$)	80 (0.61)	30 (0.61)	30 (0.61)	30 (0.61)

7. Discussion

Our work provides theoretical grounding for arbitrarily slowing convergence rates under non-adversarial first-order (second in Section J) methods common in foundation model fine-tuning. We derive a general class of convergence-rate control methods (spectral reparameterization) and provide an illustrative example (SpecDef) that is practically effective in a non-adversarial regime. However, we also show that adaptive attackers can undo these resistance methods, albeit with a linear increase in budget cost.

Practical impact The utility of our results depends on whether one seeks *safety* guarantees (settings without adversaries) or *security* guarantees (assuming adversaries). Geometric safety constraints in open-weight models are practically useful in the safety regime—for example by making accidental removal of safety guards more difficult (Qi et al., 2023) or protecting against reward-incentivized unsafe behaviour (Dai et al., 2024) i.e., "reward hacking". One limitation of our method is that it prevents training on all distributions but our framework provides a foundation for remedying this. For example, future work could localize convergence results by aligning distributions with specific singular values (Zheng et al., 2025), creating distribution-specific training constraints.

Security A main contribution of this work is *provably* ruling out convergence-rate control as a path to security guarantees on open-weight models. Future work could explore means of making k-th layer injection even more expensive, for example, by layering in defences such as gradient obfuscation. However, even requiring a substantial training budget only deters low-skilled and low-resourced attackers. These constraints would not prevent state actors developing weapons motivated to undo safety guards. Open-weight models also have a large inference-time attack surface (Rando et al., 2025) as well as vulnerability to model exfiltration, such as through distillation (Dionicio et al., 2025) that our work does not address. Future work should explore applications of traditional security solutions such as hardware attestation, functional encryption, and model compilation, which could provide future pathways towards security guarantees.

Structural insights for neural network optimization We develop new bounds linking Hessian spectral values to parameter matrix spectral values in deep networks. Developing this bound further could provide new viable alternatives to approaches for curvature control and analysis based on

Hessian-vector products (e.g., Wang et al., 2025a).

8. Conclusion

Our work establishes a concrete theoretical foundation for training-time safety. Theorem 3 enables tractable curvature control at scale, and SpecDef provides resistance where all tested prior methods fail (Table 1), including against second-order optimizers (Table 7). Theorems 8 and 7 show that training-time safety mechanisms relying solely on optimization dynamics are inherently limited. While they can impose meaningful barriers for compute-limited attackers and suggest new avenues for guaranteed safety in non-adversarial settings, they offer little protection against well-resourced adversaries, sharply delineating the limits of what is possible in open-weight model safety.

Acknowledgements

We would like to acknowledge the generous support of the Killam Foundation, the Vector Institute of Artificial Intelligence, and the Natural Sciences and Engineering Research Council of Canada for funding this work. The compute was made available by the Digital Research Alliance of Canada, Vector Institute, and a grant from the Center for AI Safety.

Impact statement

This paper establishes fundamental limits that may narrow the space of viable open-weight safety approaches. Clearly understanding what is and is not possible is essential for the field's maturity. Impossibility results guide future research toward promising directions (cryptographic methods, architectural interventions) while avoiding dead ends. By providing a theoretical basis for understanding resistance mechanisms and their limitations, we demonstrate significant progress in training-time safety. We hope this paper influences the community to approach this topic from first principles and sets a standard for formal theoretical guarantees in future open-weight safety methods.

References

Abdalla, A., Shaheen, I., DeGenaro, D., Mallick, R., Raita, B., and Bargal, S. A. Gift: Gradient-aware immunization of diffusion models against malicious fine-tuning with safe concepts retention. *arXiv preprint arXiv:2507.13598*, 2025. pages 32

Anderljung, M., Barnhart, J., Korinek, A., Leung, J., O'Keefe, C., Whittlestone, J., Avin, S., Brundage, M., Bullock, J., Cass-Beggs, D., et al. Frontier ai regulation: Managing emerging risks to public safety. *arXiv preprint arXiv:2307.03718*, 2023. pages 1

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018. pages 3, 43

Bach, F. *Learning theory from first principles*. MIT press, 2024. pages 14

Baskerville, N. P., Granziol, D., and Keating, J. P. Appearance of random matrix theory in deep learning. *Physica A: Statistical Mechanics and its Applications*, 590:126742, 2022a. pages 2, 50

Baskerville, N. P., Keating, J. P., Mezzadri, F., Najnudel, J., and Granziol, D. Universal characteristics of deep neural network loss surfaces from random matrix theory. *Journal of Physics A: Mathematical and Theoretical*, 55 (49):494002, 2022b. pages 2

Bhatia, R. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013. pages 3, 16, 17

Bubeck, S. and Mikulincer, D. How to trap a gradient flow, 2020. URL https://arxiv.org/abs/2001.02968. pages 2

Casper, S., O'Brien, K., Longpre, S., Seger, E., Klyman, K., Bommasani, R., Nrusimha, A., Shumailov, I., Mindermann, S., Basart, S., et al. Open technical problems in open-weight ai model risk management. *Social Science Research Network*, 2025. pages 1

Che, Z., Casper, S., Kirk, R., Satheesh, A., Slocum, S., McKinney, L. E., Gandikota, R., Ewart, A., Rosati, D., Wu, Z., Cai, Z., Chughtai, B., Gal, Y., Huang, F., and Hadfield-Menell, D. Model tampering attacks enable more rigorous evaluations of LLM capabilities. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=E60YbLnQd2. pages 1, 6, 30, 31

Cheng, Z., Zhang, M., Sun, J., and Dai, W. On weaponization-resistant large language models with prospect theoretic alignment. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 10309–10324, 2025. pages 31

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pp. 192–204. PMLR, 2015. pages 2

Dai, J., Pan, X., Sun, R., Ji, J., Xu, X., Liu, M., Wang, Y., and Yang, Y. Safe RLHF: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TyFrPOKYXw. pages 8

Davis, C. and Kahan, W. M. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970. pages 20

De, S., Mukherjee, A., and Ullah, E. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018. pages 2

Deeb, A. and Roger, F. Do unlearning methods remove information from language model weights? *arXiv preprint arXiv:2410.08827*, 2024. pages 29

Défossez, A., Bottou, L., Bach, F., and Usunier, N. A simple convergence proof of adam and adagrad. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. pages 2

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017. pages 4

Dionicio, S., Elahi, A., Rosati, D., and Sajjad, H. Undistillable open language models with teacher scrambling. In *Lock-LLM Workshop: Prevent Unauthorized Knowledge Use from Large Language Models*, 2025. URL https://openreview.net/forum?id=g9vFg3O8YY. pages 8

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pp. 1675–1685. PMLR, 2019. pages 7, 48

Fan, C., Liu, J., Lin, L., Jia, J., Zhang, R., Mei, S., and Liu, S. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. In *Neurips Safe Generative AI Workshop 2024*, 2024. pages 30

Fan, C., Jia, J., Zhang, Y., Ramakrishna, A., Hong, M., and Liu, S. Towards llm unlearning resilient to relearning attacks: A sharpness-aware minimization perspective and beyond, 2025. URL https://arxiv.org/abs/2502.05374. pages 6, 30

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020. pages 4

Gandikota, R., Materzyńska, J., Fiotto-Kaufman, J., and Bau, D. Erasing concepts from diffusion models. In *Proceedings of the 2023 IEEE International Conference on Computer Vision*, 2023. pages 35

Gandikota, R., Feucht, S., Marks, S., and Bau, D. Erasing conceptual knowledge from language models. *arXiv preprint arXiv:2410.02760*, 2024. pages 30

Goh, G. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006. URL http://distill.pub/2017/momentum. pages 15

Golatkar, A., Achille, A., and Soatto, S. Eternal sunshine of the spotless net: Selective forgetting in deep networks. in 2020 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9301–9309, 2019. pages 2, 30

Henderson, P., Mitchell, E., Manning, C., Jurafsky, D., and Finn, C. Self-destructing models: Increasing the costs of harmful dual uses of foundation models. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 287–296, 2023. pages 5, 29

Hollender, A. and Zampetakis, E. The computational complexity of finding stationary points in non-convex optimization. In Neu, G. and Rosasco, L. (eds.), *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pp. 5571–5572. PMLR, 12–15 Jul 2023. URL https://proceedings.mlr.press/v195/hollender23a.html. pages 2

Horn, R. A. and Johnson, C. R. *Topics in Matrix Analysis*. Cambridge University Press, 1991. pages 3, 16, 17

Huang, T., Hu, S., Ilhan, F., Tekin, S. F., and Liu, L. Harmful fine-tuning attacks and defenses for large language models: A survey. *arXiv preprint arXiv:2409.18169*, 2024a. pages 1, 31

Huang, T., Hu, S., and Liu, L. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. *Advances in Neural Information Processing Systems*, 37:74058–74088, 2024b. pages 31

Huanjian, Z., Andi, H., Akiko, T., and Masashi, S. The adaptive complexity of finding a stationary point. In *The Thirty Eighth Annual Conference on Learning Theory*, pp. 6091–6123. PMLR, 2025. pages 2

Inan, H., Upasani, K., Chi, J., Rungta, R., Iyer, K., Mao, Y., Tontchev, M., Hu, Q., Fuller, B., Testuggine, D., and Khabsa, M. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. URL https://arxiv.org/abs/2312.06674. pages 31

Ipsen, I. C. and Meyer, C. D. The angle between complementary subspaces. *The American mathematical monthly*, 102(10):904–911, 1995. pages 20

Ji, J., Liu, M., Dai, J., Pan, X., Zhang, C., Bian, C., Zhang, C., Sun, R., Wang, Y., and Yang, Y. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *arXiv preprint arXiv:2307.04657*, 2023. pages 31

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., and Bernstein, J. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. pages 2, 6, 53

Kaur, A. and Lui, S. New lower bounds on the minimum singular value of a matrix. *Linear Algebra and its Applications*, 666:62–95, 2023. pages 20

Knyazev, A. V. and Zhu, P. Principal angles between subspaces and their tangents. *arXiv preprint arXiv:1209.0523*, 2012. pages 20

Kristiadi, A., Dangel, F., and Hennig, P. The geometry of neural nets' parameter spaces under reparametrization. *Advances in Neural Information Processing Systems*, 36: 17669–17688, 2023. pages 4

Lattimore, T. and Szepesvári, C. *Bandit algorithms*. Cambridge University Press, 2020. pages 29

Li, J. and Hong, M. A note on the convergence of muon. *arXiv preprint arXiv:2502.02900*, 2025. pages 53

Li, N., Pan, A., Gopal, A., Yue, S., Berrios, D., Gatti, A., Li, J. D., Dombrowski, A.-K., Goel, S., Phan, L., Mukobi, G., Helm-Burger, N., Lababidi, R., Justen, L., Liu, A. B., Chen, M., Barrass, I., Zhang, O., Zhu, X., Tamirisa, R., Bharathi, B., Khoja, A., Zhao, Z., Herbert-Voss, A., Breuer, C. B., Marks, S., Patel, O., Zou, A., Mazeika, M., Wang, Z., Oswal, P., Liu, W., Hunt, A. A., Tienken-Harder, J., Shih, K. Y., Talley, K., Guan, J., Kaplan, R., Steneker, I., Campbell, D., Jokubaitis, B., Levinson, A., Wang, J., Qian, W., Karmakar, K. K., Basart, S., Fitz, S., Levine, M., Kumaraguru, P., Tupakula, U., Varadharajan, V., Shoshitaishvili, Y., Ba, J., Esvelt, K. M., Wang, A., and Hendrycks, D. The wmdp benchmark: Measuring and reducing malicious use with unlearning, 2024. pages 5, 29, 30

Liu, B., Liu, Q., and Stone, P. Continual learning and private unlearning. In *Conference on Lifelong Learning Agents*, pp. 243–254. PMLR, 2022. pages 30

Liu, H., Li, Z., Hall, D. L. W., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2023. pages 2, 6, 48

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. pages 29

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. pages 29

Miao, J. and Ben-Israel, A. On principal angles between subspaces in rn. *Linear algebra and its applications*, 171: 81–98, 1992. pages 20

Mirzadeh, I., Alizadeh, K., Mehta, S., Del Mundo, C. C., Tuzel, O., Samei, G., Rastegari, M., and Farajtabar, M. Relu strikes back: Exploiting activation sparsity in large language models. *arXiv preprint arXiv:2310.04564*, 2023. pages 4

Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Dokl. Akad. Nauk. SSSR*, volume 269, pp. 543, 1983. pages 15

Nesterov, Y. *Introductory lectures on convex optimization*. Springer Nature, 2013. pages 2

Nevo, S., Lahav, D., Karpur, A., Bar-On, Y., Bradley, H.-A., and Alstott, J. *Securing AI model weights: Preventing theft and misuse of frontier models*. Rand Corporation, 2024. pages 1

Neyshabur, B., Sedghi, H., and Zhang, C. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020. pages 2

O'Brien, K., Casper, S., Anthony, Q., Korbak, T., Kirk, R., Davies, X., Mishra, I., Irving, G., Gal, Y., and Biderman, S. Deep ignorance: Filtering pretraining data builds tamper-resistant safeguards into open-weight llms. *arXiv preprint arXiv:2508.06601*, 2025. pages 6, 30, 43

OpenAI, :, Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., Barak, B., Bennett, A., Bertao, T., Brett, N., Brevdo, E., Brockman, G., Bubeck, S., Chang, C., Chen, K., Chen, M., Cheung, E., Clark, A., Cook, D., Dukhan, M., Dvorak, C., Fives, K., Fomenko, V., Garipov, T., Georgiev, K., Glaese, M., Gogineni, T., Goucher, A., Gross, L., Guzman, K. G., Hallman, J., Hehir, J., Heidecke, J., Helyar, A., Hu, H., Huet, R., Huh, J., Jain, S., Johnson, Z., Koch, C., Kofman, I., Kundel, D., Kwon, J., Kyrylov, V., Le, E. Y., Leclerc, G., Lennon, J. P., Lessans, S., Lezcano-Casado, M., Li, Y., Li, Z., Lin, J., Liss, J., Lily, Liu, Liu, J., Lu, K., Lu, C., Martinovic, Z., McCallum, L., McGrath, J., McKinney, S., McLaughlin, A., Mei, S., Mostovoy, S., Mu, T., Myles, G., Neitz, A., Nichol, A., Pachocki, J., Paino, A., Palmie, D., Pantuliano, A., Parascandolo, G., Park, J., Pathak, L., Paz, C., Peran, L., Pimenov, D., Pokrass, M., Proehl, E., Qiu, H., Raila, G., Raso, F., Ren, H., Richardson, K., Robinson, D., Rotsted, B., Salman, H., Sanjeev, S., Schwarzer, M., Sculley, D., Sikchi, H., Simon, K., Singhal, K., Song, Y., Stuckey, D., Sun, Z., Tillet, P., Toizer, S., Tsimpourlas, F., Vyas, N., Wallace, E., Wang, X., Wang,

M., Watkins, O., Weil, K., Wendling, A., Whinnery, K., Whitney, C., Wong, H., Yang, L., Yang, Y., Yasunaga, M., Ying, K., Zaremba, W., Zhan, W., Zhang, C., Zhang, B., Zhang, E., and Zhao, S. gpt-oss model card, 2025. URL https://arxiv.org/abs/2508.10925. pages 32

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964. pages 15

Qi, X., Zeng, Y., Xie, T., Chen, P.-Y., Jia, R., Mittal, P., and Henderson, P. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023. pages 1, 8

Qi, X., Wei, B., Carlini, N., Huang, Y., Xie, T., He, L., Jagielski, M., Nasr, M., Mittal, P., and Henderson, P. On evaluating the durability of safeguards for open-weight llms. *arXiv preprint arXiv:2412.07097*, 2024. pages 1, 29, 30

Raginsky, M., Rakhlin, A., and Telgarsky, M. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pp. 1674–1703. PMLR, 2017. pages 2

Rando, J., Zhang, J., Carlini, N., and Tramèr, F. Adversarial ml problems are getting harder to solve and to evaluate. *arXiv preprint arXiv:2502.02260*, 2025. pages 8

Rosati, D., Wehner, J., Williams, K., Bartoszcze, L., Gonzales, R., Majumdar, S., Sajjad, H., Rudzicz, F., et al. Representation noising: A defence mechanism against harmful finetuning. *Advances in Neural Information Processing Systems*, 37:12636–12676, 2024a. pages 6, 30, 31, 42

Rosati, D., Wehner, J., Williams, K., Bartoszcze, L., Sajjad, H., and Rudzicz, F. Immunization against harmful fine-tuning attacks. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5234–5247, Miami, Florida, USA, November 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.301. URL https://aclanthology.org/2024.findings-emnlp.301/. pages 1, 2, 41

Schramowski, P., Brack, M., Deiseroth, B., and Kersting, K. Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. pages 32

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 07 2002. ISSN 0899-7667. doi: 10.1162/08997660260028683. URL https://doi.org/10.1162/08997660260028683. pages 25

Seger, E., Dreksler, N., Moulange, R., Dardaman, E., Schuett, J., Wei, K., Winter, C., Arnold, M., hÉigeartaigh, S. Ó., Korinek, A., et al. Open-sourcing highly capable foundation models: An evaluation of risks, benefits, and alternative methods for pursuing open-source objectives. *arXiv preprint arXiv:2311.09227*, 2023. pages 1

Shilov, I., Cloud, A., Gema, A. P., Goldman-Wetzler, J., Panickssery, N., Sleight, H., Jones, E., and Anil, C. Beyond data filtering: Knowledge localization for capability removal in llms. *arXiv preprint arXiv:2512.05648*, 2025. pages 30

Simsek, B., Ged, F., Jacot, A., Spadaro, F., Hongler, C., Gerstner, W., and Brea, J. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pp. 9722–9732. PMLR, 2021. pages 4

Sondej, F. and Yang, Y. Collapse of irrelevant representations (cir) ensures robust and non-disruptive llm unlearning. *arXiv preprint arXiv:2509.11816*, 2025. pages 30

Tamirisa, R., Bharathi, B., Phan, L., Zhou, A., Gatti, A., Suresh, T., Lin, M., Wang, J., Wang, R., Arel, R., Zou, A., Song, D., Li, B., Hendrycks, D., and Mazeika, M. Tamper-resistant safeguards for open-weight llms, 2025. URL https://arxiv.org/abs/2408.00761. pages 1, 6, 29, 30, 31, 41

Tzen, B., Liang, T., and Raginsky, M. Local optimality and generalization guarantees for the langevin algorithm via empirical metastability. In *Conference On Learning Theory*, pp. 857–875. PMLR, 2018. pages 2

Wallace, E., Watkins, O., Wang, M., Chen, K., and Koch, C. Estimating worst-case frontier risks of open-weight llms. *arXiv preprint arXiv:2508.03153*, 2025. pages 1

Wang, A., Nguyen, E., Yang, R., Bae, J., McIlraith, S. A., and Grosse, R. Better training data attribution via better inverse hessian-vector products. *arXiv preprint arXiv:2507.14740*, 2025a. pages 9

Wang, Y., Zhu, R., and Wang, T. Self-destructive language model. *arXiv preprint arXiv:2505.12186*, 2025b. pages 31

Wright, S. J. and Recht, B. *Optimization for Data Analysis*. Cambridge University Press, 2022. pages 2, 14, 15, 16

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer

normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020. pages 7, 22, 48

Xu, J., Sun, X., Zhang, Z., Zhao, G., and Lin, J. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32, 2019. pages 22

Yao, Z., Gholami, A., Shen, S., Mustafa, M., Keutzer, K., and Mahoney, M. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021. pages 2, 6, 48

Yi, B., Huang, T., Zhang, B., Li, T., Nie, L., Liu, Z., and Shen, L. Ctrap: Embedding collapse trap to safeguard large language models from harmful fine-tuning. *arXiv preprint arXiv:2505.16559*, 2025. pages 31

Zhang, R., Lin, L., Bai, Y., and Mei, S. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024. pages 30

Zhang, X. *Notes for Optimization Algorithms*. Department of Mathematics, Purdue University, 2023. URL https://www.math.purdue.edu/~zhan1966/teaching/598/Opt_notes.pdf. Spring 2023 lecture notes. pages 54, 55

Zhang, Y., Guo, X., and Xie, P. A note on perturbation analysis for t-product based tensor singular values. *arXiv preprint arXiv:2109.11194*, 2021. pages 54, 56, 57, 58, 59

Zhao, B., Dehmamy, N., Walters, R., and Yu, R. Symmetry teleportation for accelerated optimization. *Advances in neural information processing systems*, 35:16679–16690, 2022. pages 4

Zheng, A. Y. and Yeh, R. A. Imma: Immunizing text-to-image models against malicious adaptation. In *European Conference on Computer Vision*, pp. 458–475. Springer, 2024. pages 1, 6, 30, 32, 35, 39

Zheng, A. Y., Bai, C. S., Bullins, B., and Yeh, R. A. Model immunization from a condition number perspective. *arXiv preprint arXiv:2505.23760*, 2025. pages 8

Zou, A., Phan, L., Wang, J., Duenas, D., Lin, M., Andriushchenko, M., Kolter, J. Z., Fredrikson, M., and Hendrycks, D. Improving alignment and robustness with circuit breakers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2023. pages 30

A. Review of First-Order Methods

This section reviews convergence rates of first-order (gradient-based) steepest descent methods. A deeper review appears in Wright & Recht (2022). In foundation model fine-tuning, we minimize a loss $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$ over parameters $\theta \in \mathbb{R}^p$ (the concatenation of all weight matrices) where predictions are made by a prediction function $f_\theta : \mathbb{R}^d \to \mathbb{R}$ for a dataset $\mathcal{D} = \{(x_j \in \mathbb{R}^d, y_j \in \mathbb{R})\}_{j=1}^n$ with n samples and some convex loss like mean-squared error $\|f_\theta(x_i) - y_i\|_2^2$ defines the loss $\mathcal{L}$. For simplicity in this analysis we will treat full batch gradient descent. Under common assumptions like i.i.d. sampling, empirical risk minimization converges to the population risk as n increases (Bach, 2024). Gradient descent minimizes $\mathcal{L}$ via the recurrence relation:

$$\theta_{k+1} = \theta_k - \eta \nabla \mathcal{L}_{\theta_k}, \quad k = 0, 1, 2, \ldots$$

for learning rate $\eta \in \mathbb{R}_+$. First-order optimality conditions state that when the gradient achieves $\nabla \mathcal{L}_\theta = 0$ a stationary point θ_* is found. Under assumptions below, Gradient descent finds such points with appropriate η (proof below). The convergence rate specifies iterations k required to reach a stationary point. We assume $\mathcal{L}$ is L-smooth (Theorem 9), standard in optimization theory and deep learning. Sobolev-type regularity handles non-smooth components (e.g., weak differentiability of ReLU), but incorporating these would complicate exposition without changing overall conclusions of this paper.

Definition 9 (L-smooth functions)**.** A function f is said to have L-Lipschitz continuous gradients, or be L-smooth, if, with appropriate norm $\|\cdot\|$, for all x and y,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

Application of the L-smooth property to a Taylor series approximation of $\mathcal{L}$ yields the following well-known descent lemma.

Lemma 10 (Descent Lemma)**.** *For the gradient descent update rule of L-smooth functions, we have the following upper bound*

$$\mathcal{L}_{\theta_{k+1}} \leq \mathcal{L}_{\theta_k} - \frac{1}{2L}\|\nabla \mathcal{L}_{\theta_k}\|^2.$$

The full proof appears in Wright & Recht (2022) and involves minimizing an L-smooth Taylor approximation of $\mathcal{L}$. The optimal learning rate is $\eta = 1/L$—the maximal rate before oscillation ($1/L < \eta \leq 2/L$) or divergence ($\eta > 2/L$). We assume convergent behaviour, as otherwise optimization never finds an stationary point. This justifies our assumption throughout that learning rate is at most $1/L$.

A.1. Convergence Rate of GD on Smooth Functions

We can use Theorem 10 to derive the rate of convergence of gradient descent on smooth functions.

Summing over the descent lemma from iteration $i = 0$ to $k - 1$ results in

$$\mathcal{L}_{\theta_k} \leq \mathcal{L}_{\theta_0} - \frac{1}{2L}\sum_{i=0}^{k-1}\|\nabla \mathcal{L}_{\theta_i}\|^2.$$

Assuming $\mathcal{L}$ has a local minimum $\mathcal{L}_{\theta_*}$ such that $\mathcal{L}_{\theta_*} \leq \mathcal{L}_{\theta_k}$ for all k in some neighbourhood of θ_*, then we have

$$\sum_{i=0}^{k-1}\|\nabla \mathcal{L}_{\theta_i}\|^2 \leq 2L\left[\mathcal{L}_{\theta_0} - \mathcal{L}_{\theta_*}\right],$$

which implies that $\lim_{k\to\infty}\|\nabla \mathcal{L}_{\theta_k}\| = 0$ otherwise the upper bound could not hold. Finally we have:

$$\min_{0\leq i\leq k-1}\|\nabla \mathcal{L}_{\theta_i}\|^2 \leq \frac{1}{k}\sum_{i=0}^{k-1}\|\nabla \mathcal{L}_{\theta_i}\|^2 \leq \frac{2L\left[\mathcal{L}_{\theta_0} - \mathcal{L}_{\theta_*}\right]}{k},$$

which follows from the fact that the minimum of a function is always less than or equal its mean.

This formula shows that after k steps of gradient descent, the minimum gradient norm shrinks as

$$\min_{0\leq i\leq k-1}\|\nabla \mathcal{L}_{\theta_i}\| \leq \sqrt{\frac{2L\left[\mathcal{L}_{\theta_0} - \mathcal{L}_{\theta_*}\right]}{k}}.$$

We can restate this using another property that follows from the Taylor approximation of L-smooth functions that $\mathcal{L}_{\theta_0} - \mathcal{L}_{\theta_*} \leq \frac{L}{2}\|\theta_0 - \theta_*\|^2$ resulting in

$$\min_{0\leq i\leq k-1} \|\nabla \mathcal{L}_{\theta_i}\| \leq \sqrt{\frac{[L\|\theta_0 - \theta_*\|]^2}{k}}.$$

A.2. Iteration Complexity

From this we can show the iteration complexity of Section A.1. The iteration complexity tells us how many steps k are needed to achieve $\min_{0\leq i\leq k-1} \|\nabla \mathcal{L}_{\theta_i}\| \leq \epsilon$. We can show the iteration complexity is given by Theorem 1 from the main text with the following proof. The proposition is restated for clarity here first.

Proposition 11 (L-smooth gradient descent iteration complexity)**.** *Let $\mathcal{L}$ be L-smooth. For L-smooth gradient descent, the guarantee is that to obtain $\min_{0\leq i\leq k-1} \|\nabla \mathcal{L}_{\theta_i}\| \leq \epsilon$, the number of steps is given by the lower bound $k \geq [L\|\theta_0 - \theta_*\|]^2/\epsilon^2$ where θ_0 is the initial model weights, i is the sequence index, and θ_* is a stationary point. and k is the number of iteration steps.*

Proof. By directly substituting $k = \frac{1}{\epsilon^2}[L\|\theta_0 - \theta_*\|]^2$ in Section A.1 we achieve $\min_{0\leq i\leq k-1} \|\nabla \mathcal{L}_{\theta_i}\| \leq \epsilon$. Therefore, the iteration complexity of smooth GD is $k \geq [L\|\theta_0 - \theta_*\|]^2/\epsilon^2$.

□

A.3. Nesterov's Optimal Method Convergence Analysis

Nesterov's method belongs to a category of "momentum"-based methods and is often motivated as a way to "dampen" the oscillatory behaviour introduced by gradient descent. Oscillatory behaviour occurs when there are both small and large curvature directions, causing nearly orthogonal gradients (This phenomenon is illustrated in Goh, 2017). Nesterov's optimal (or accelerated) method (Nesterov, 1983) involves two modifications to the typical gradient descent method.

First, we introduce a momentum or drag term $\beta(\theta_i - \theta_{i-1})$ which prevents overshooting and yields the so-called heavy ball method or Polyak's momentum (Polyak, 1964). Second, instead of evaluating our gradient at θ_i, we evaluate the gradient at $\theta_i + \beta(\theta_i - \theta_{i-1})$ in order to correct for the path taken by momentum. This results in the following descent equations:

$$\begin{aligned}\phi_u &= \theta_i + \beta_i(\theta_i - \theta_{i-1}) \\ \theta_{i+1} &= \phi_u - \alpha\nabla_{\theta_i}\mathcal{L}(\phi_u).\end{aligned}$$

Deriving the convergence rate of Nesterov's method and why it is optimal is outside the scope of this paper, since we are simply using the known convergence rate to show that best-case function convergence still depends on L. Interested readers are referred to Wright & Recht (2022), from which we obtain the convergence rate result restated below.

$$\mathcal{L}_{\theta_k} - \mathcal{L}_{\theta_*} \leq \frac{2L}{(k+1)^2}\|\theta_0 - \theta_*\|^2. \tag{1}$$

Using this we can prove Theorem 2 from the main text (restated below for clarity).

Proposition 12 (Nesterov iteration complexity)**.** *Let $\mathcal{L}$ be an L-smooth convex function. For Nesterov's optimal method, the guarantee is that to obtain $\mathcal{L}_{\theta_k} - \mathcal{L}_{\theta_*} \leq \epsilon$, the number of steps is given by the lower bound $k \geq \sqrt{[2L\|\theta_0 - \theta_*\|^2]/\epsilon} - 1$.*

Proof. As above, our goal is to set k such that $\mathcal{L}_{\theta_k} - \mathcal{L}_{\theta_*} \leq \epsilon$. Let $\Delta = \|\theta_0 - \theta_*\|^2$. Set the L.H.S of Equation (1) to ϵ to find k.

$$\begin{aligned}\frac{2L\Delta}{(k+1)^2} &= \epsilon \\ \sqrt{2L\Delta/\epsilon} - 1 &= k.\end{aligned}$$

From above, we see that we must set k to $\sqrt{2L\Delta/\epsilon} - 1$ in order to achieve ϵ error in the L.H.S of Equation (1). Proving the iteration complexity is $k \geq \sqrt{2L\Delta/\epsilon} - 1$. □

A.4. Hessian Singular Value Dependence

The results of our paper depend on the following property.

Proposition 13. *The smoothness quantifier $L \in \mathbb{R}$ for the function $\mathcal{L}$ is greater than or equal to the maximum singular value of the Hessian H of that function. In other words the following formula holds $L \geq \sigma_1(H_\theta^{\mathcal{L}})$.*

Proof. For an L-smooth function f, smoothness is equivalent to a uniform operator-norm bound on the Hessian:

$$\|\nabla^2 f(\theta)\|_2 \leq L, \quad \text{for all } \theta$$

(see, e.g., Wright & Recht, 2022). Equivalently, for any unit vector v,

$$|v^\top \nabla^2 f(\theta)\, v| \leq L.$$

Since $\nabla^2 f(\theta)$ is symmetric, all of its eigenvalues lie in the interval $[-L, L]$. Therefore, the largest singular value of the Hessian satisfies

$$\sigma_1\big(H_\theta^{\mathcal{L}}\big) = \|H_\theta^{\mathcal{L}}\|_2 = \max_i |\lambda_i(H_\theta^{\mathcal{L}})| \leq L.$$

Equivalently, this implies the two-sided Loewner bound

$$-LI \preceq \nabla^2 f(\theta) \preceq LI,$$

which completes the proof. □

To connect this back to convergence rate, we present a classical demonstration of the impact of principal curvature (L) on number of steps needed to minimize a quadratic under gradient descent in Figure 3.

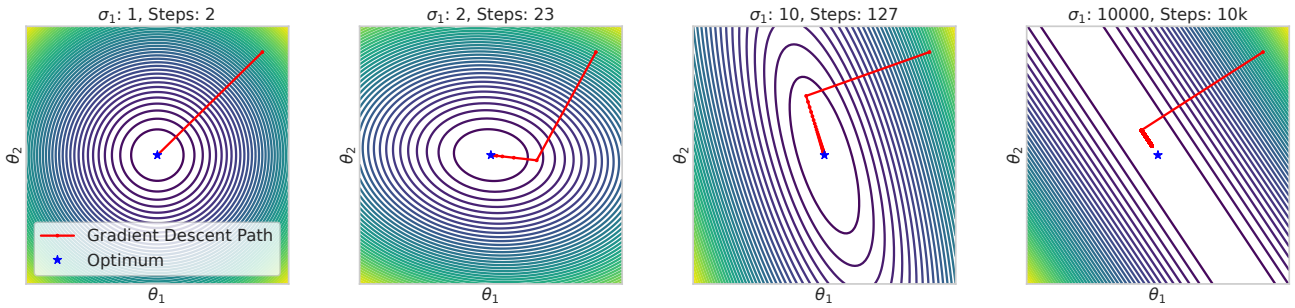


Figure 3. Steps needed for optimizing a 2D convex quadratic function depend on Hessian spectral values, σ_1, σ_2. Maximum convergent learning rate is used and σ_2 is fixed at 1.

B. Review of Singular Value Inequalities

To prepare for understanding the results of this paper and the proof techniques used, we briefly review singular value inequalities for general real matrices. Horn & Johnson (1991, Chapter 3) provides a readable overview from which much of this material is sourced ((Bhatia, 2013) is another good resource).

Recall the spectral value decomposition states that given a real-valued matrix $A \in \mathbb{R}^{m\times n}$ and index $q = \min(m, n)$, there is a diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_q)$ with $\sigma_1 \geq \cdots \geq \sigma_q$ and two orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that $A = U\Sigma V^\top$. The individual elements σ_i are the singular values of A, U is known as the left singular vectors of A and V as the right singular vectors.

The Poincaré separation Theorem for singular values allows us to characterize matrix singular values by bounding them in terms of their submatrices, where a submatrix A_r is given by deleting a total number of r rows and/or columns from a matrix $A \in \mathbb{R}^{m\times n}$. We restate this result from Horn & Johnson (1991, 3.1.3, p. 149) where a proof can be obtained. We use this result extensively to find tractably computable submatrices of the Hessian and bound its singular values.

Theorem 14 (Poincaré separation Theorem for singular values)**.** *Given $A \in \mathbb{R}^{m\times n}$ and submatrix A_r the following holds*

$$\sigma_i(A) \geq \sigma_i(A_r) \geq \sigma_{i+r}(A), \quad i = 1, \ldots, \min\{m, n\}$$

where for $X \in \mathbb{R}^{p\times q}$, let $\sigma_i \equiv 0$ if $i \geq \min\{p, q\}$.

The Courant-Fischer Theorem allows us to have a variational characterization of singular values based on the subspaces of matrices. We use this later in the paper to construct lower bounds that consider the subspace alignment between matrices A and B in matrix product AB. For now, we can consider it as an alternative way to obtain singular values given the norm of a matrix-vector product:

Theorem 15 (Courant-Fischer)**.** *Let $A \in \mathbb{R}^{m\times n}$ and $\sigma_1(A) \geq \sigma_2(A) \geq \ldots$ be the ordered singular values of A for all $1 \leq i \leq \min\{m, n\}$. Then*

$$\begin{aligned}\sigma_i(A) = &\max_{\dim S=i} \min_{\substack{x\in S \\ \|x\|_2=1}} \|Ax\|_2 \\ = &\min_{\dim S=n-i+1} \max_{\substack{x\in S \\ \|x\|_2=1}} \|Ax\|_2.\end{aligned}$$

$\mathcal{S}$ are the subspaces of $\mathbb{R}^n$ with the indicated dimension i.

The proof of this is stated in Horn & Johnson (1991, 3.1.2).

We now state two-sided inequalities for products and sums of matrices that we need for decomposing Hessian block (i.e., $\frac{\partial^2 \mathcal{L}}{\partial \theta_i \partial \theta_k}$) formulas.

Theorem 16 (Additive Singular Value Bounds)**.** *Let $A, B \in \mathbb{R}^{m\times n}$ be given and let $q = \min\{m, n\}$. The following inequality holds for decreasingly ordered singular values of A, B, and A + B.*

$$\sigma_i(A) - \sigma_1(B) \leq \sigma_i(A + B) \leq \sigma_i(A) + \sigma_1(B).$$

For $1 \leq i \leq q$.

The proof of this and the Theorem below are presented in Horn & Johnson (1991, Th 3.3.16).

Theorem 17 (Multiplicative Singular Value Bounds)**.** *Let $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{m\times p}$ and let $q = \min\{n, p\}$. The following inequality holds for decreasingly ordered singular values of A, B, and AB.*

$$\sigma_i(A)\sigma_q(B) \leq \sigma_i(AB) \leq \sigma_i(A)\sigma_1(B).$$

For $1 \leq i \leq q$

Note that the bound for Theorem 17 is symmetric, that is $\sigma_q(A)\sigma_i(B) \leq \sigma_i(AB) \leq \sigma_1(A)\sigma_i(B)$ also holds.

Tighter upper bounds can be derived from majorization results such as those of Lidskii and Ky Fan, but the focus of our paper is largely on lower bounds. Additional characterization can be obtained probabilistically using random matrix theory, but this is outside the scope of this paper. Unfortunately, the lower bounds presented can be vacuous for rank-deficient matrices, which are common in deep learning, and are generally uninformative for small singular values. There are a number of ways to remedy this, including Gel'fand-Naimark's results in Bhatia (2013, III.20), but these require strong assumptions (e.g., positive definiteness in the case of Gel'fand-Naimark). To address this, we develop a stronger result without additional assumptions in Section C.

C. Principal Angle Lower Bound

Now that we have provided a review of singular value inequalities in Section B, we develop a tighter lower bound that depends on the overlap between the image of B and the span of A's left singular vectors. We develop this result because, in our setting, the Hessian blocks often factor into components that the designer cannot control (A), namely the data distribution or architectural components, and those the designer can control (B), for example rotations of θ_i that change the image of B and scaling that changes the singular values.

Before stating this result, let us review subspaces, principal angles, orthogonal projections, and images. The image, or range, of a linear transformation represented by a matrix $B \in \mathbb{R}^{m\times n}$ and denoted $\mathrm{Im}(B)$ is the span of its column vectors, that is $\mathrm{Im}(B) := \mathrm{span}(B_{:,1}, \ldots, B_{:,n})$. To make things easier for later definitions, recall that by the singular value decomposition any matrix B can be written as $B = U_B \Sigma_B V_B^\top$ with orthogonal $U_B \in \mathbb{R}^{m\times m}$ and $V_B \in \mathbb{R}^{n\times n}$ singular vector matrices. The image of B is spanned by the first r columns of the left singular vectors $\mathrm{Im}(B) := \mathrm{span}(U_{B_{:,1}}, \ldots, U_{B_{:,r}})$ where $r = \mathrm{rank}(B)$, a fact we use later. A subspace $\mathcal{M} \subset \mathcal{V}$ is a subset that is closed under linear combination of its vectors. The dimension of this subspace is the number of elements in the basis of $\mathcal{M}$, i.e., the linearly independent vectors that span $\mathcal{M}$. To illustrate, a rank-k subspace of $\mathrm{Im}(B)$ can be represented by the first k left singular vectors of B denoted U_{B_k}. An orthogonal projection Π_E is a linear transformation that takes any vector in the ambient space onto a subspace spanned by E. If $U_E \in \mathbb{R}^{m\times r}$ is the left singular vector matrix corresponding to nonzero singular values of E, then $\Pi_E = U_E U_E^\top$ (since U_E is orthogonal, the term $(U_E^\top U_E)^{-1}$ in a standard projection drops). We use Π_{U_k}, which is a projection onto the rank-k subspace spanned by the columns of U_k.

Finally, we define a distance between two subspaces using their **principal angles** (also called **canonical angles**). We illustrate with subspaces arising from a matrix product. Let $A \in \mathbb{R}^{p\times n}$ and $B \in \mathbb{R}^{n\times q}$, and let $\mathcal{A} \subseteq \mathbb{R}^n$ and $\mathcal{B} \subseteq \mathbb{R}^n$ be the subspaces spanned by the rows of A and the columns of B, respectively. Both subspaces live in the same ambient space $\mathbb{R}^n$, which is precisely the space where the product AB "contracts."

Let $r_A = \dim \mathcal{A} \leq \min\{p, n\}$ and $r_B = \dim \mathcal{B} \leq \min\{n, q\}$. Let $V_A \in \mathbb{R}^{n\times r_A}$ and $U_B \in \mathbb{R}^{n\times r_B}$ be matrices whose columns form orthonormal bases for $\mathcal{A}$ and $\mathcal{B}$ (e.g., obtained from the SVD of $A^\top$ and B, respectively).

Definition 18. (Principal Angle) The **principal angles** $\theta_1, \ldots, \theta_r$ between $\mathcal{A}$ and $\mathcal{B}$, where $r = \min\{r_A, r_B\}$, are defined by

$$\theta_i = \cos^{-1}(\sigma_i),$$

where $\sigma_1 \geq \cdots \geq \sigma_r \geq 0$ are the singular values of $V_A^\top U_B \in \mathbb{R}^{r_A \times r_B}$. These angles quantify how the row space of A and column space of B align, which directly affects the singular values of the product AB.

There are several ways to utilize principal angles. Since we are going to construct lower bounds, we are concerned with the worst-case alignment (largest angle) between the subspaces $\mathcal{A}$ and $\mathcal{B}$; this would be $\sigma_{min}(V_A^\top U_B)$. We can further use a variational characterization to analyze the angle in terms of a minimization of a projection (which we need for a proof below). In this work, we use the cosine of the largest principal angle denoted $\cos\theta(\cdot,\cdot) \in \mathbb{R}_+$.

$$\begin{aligned}
\cos\theta(A, B) &= \sigma_{\min}(V_A^\top U_B) \\
&= \min_{\|c\|=1} \|V_A V_A^\top U_B\, c\| && \text{(Variational Definition)} \\
&= \min_{\|c\|=1} \|\Pi_{V_A} U_B\, c\| && \text{(Definition of Projection)} \\
&= \min_{y\in \mathrm{Im}(U_B), \|y\|=1} \|\Pi_{V_A}\, y\|. && \text{(Change of Variable)}
\end{aligned}$$

While this characterization is somewhat more complex, we present it here to clarify a step we make in the proof below. There is only one more slight change in our notation to fully define the largest principal subspace angle: we consider the angle between V_{A_r} truncated to a rank-r subspace of A and U_{B_k} truncated to a rank-k subspace of B. We are now ready to state and prove our presented on the next page.

Theorem 19 (Principal angle-weighted singular value bound)**.** *Let $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$. Let V_{A_r} be the top-r right singular-vector subspace of A. Let $\Pi_{V_{A_r}}$ denote the orthogonal projection onto the top-r right singular subspace V_{A_r} of A. Define*

$$\cos\theta(A_r, B_k) := \min_{y\in \mathrm{Im}(U_{B_k}),\ \|y\|=1} \|\Pi_{V_{A_r}} y\| \Leftrightarrow \sigma_{\min}(V_{A_r}^\top U_{B_k})$$

Where U_{B_k} is the top-k left singular vectors of B. Then for every k with $1 \le k \le \mathrm{rank}(B)$ and $r \ge k$ we have

$$\sigma_k(AB) \ge \cos\theta(A_r, B_k)\sigma_r(A)\sigma_k(B).$$

Proof. Let $V_{B_k} \subset \mathbb{R}^p$ denote the span of the top-k right singular vectors of B. By the Courant-Fischer (Theorem 15) variational characterization for the k-th singular value of AB,

$$\sigma_k(AB) = \max_{\substack{S\subset\mathbb{R}^p \\ \dim S = k}} \ \min_{\substack{x\in S \\ \|x\|=1}} \|ABx\|.$$

Since the maximum over all k-subspaces is *at least* the value on any *particular* k-subspace, we may choose $S = V_{B_k}$ to obtain

$$\sigma_k(AB) \ge \min_{\substack{x\in V_{B_k} \\ \|x\|=1}} \|ABx\|.$$

Fix any unit $x \in V_{B_k}$. By properties of the SVD of B,

$$\|Bx\| \ge \sigma_k(B).$$

Define the unit vector $y := \frac{Bx}{\|Bx\|} \in \mathrm{Im}(U_{B_k})$. Then

$$\|ABx\| = \|A(Bx)\| = \|Bx\| \cdot \|Ay\| \ge \sigma_k(B) \cdot \|Ay\|. \tag{2}$$

Expand Ay in the right-singular basis of A. Since the squared norm satisfies

$$\begin{aligned}
\|Ay\|^2 &= \|U\Sigma V^\top y\|^2 = \|\Sigma V^\top y\|^2 && \text{(Unitary Invariance)} \\
&= \sum_{i=1}^{n} \sigma_i(A)^2 |v_i^\top y|^2 \ge \sum_{i=1}^{r} \sigma_i(A)^2 |v_i^\top y|^2 && \text{(Rank reduction to } V_{A_r}\text{)} \\
&\ge \sigma_r(A)^2 \sum_{i=1}^{r} |v_i^\top y|^2 = \sigma_r(A)^2 \|\Pi_{V_{A_r}} y\|^2,
\end{aligned}$$

we obtain

$$\|Ay\| \ge \sigma_r(A)\|\Pi_{V_{A_r}} y\|. \tag{3}$$

Combining inequalities Equation (2) and Equation (3) gives, for every $x \in V_{B_k}$,

$$\|ABx\| \ge \sigma_k(B)\sigma_r(A)\|\Pi_{V_{A_r}} y\|.$$

By definition of $\cos\theta(A_r, B_k)$ we have $\|\Pi_{V_{A_r}} y\| \ge \cos\theta(A_r, B_k)$ for all unit $y \in \mathrm{Im}(U_{B_k})$. Hence

$$\|ABx\| \ge \sigma_k(B)\sigma_r(A)\cos\theta(A_r, B_k).$$

Taking the minimum over all unit $x \in V_{B_k}$ and using the earlier Courant-Fischer inequality Section C yields

$$\sigma_k(AB) \ge \sigma_k(B)\sigma_r(A)\cos\theta(A_r, B_k),$$

which is the desired bound. □

C.1. Bound Tightness and Characterization

Remark 20 (Principal Alignment Intuition). Our lower bound captures how the geometric alignment between subspaces of A and B impacts the singular values of the product AB.

This results in the following bound characteristics:

1. When $\cos\theta(A_r, B_k) = 1$: the subspaces have perfect alignment in at least one direction, resulting in very tight bounds.
2. When $\cos\theta(A_r, B_k) = 0$ there exists at least one unit vector in $\mathrm{Im}(U_{B_k})$ whose projection onto V_A is zero i.e., at least one orthogonal direction such that the lower bound becomes trivial.
3. When $0 < \cos\theta(A_r, B_k) < 1$: we have partial alignment where meaningful lower bounds can be obtained.

Figure 4 below illustrates three examples and provides an intuitive characterization of our principal angle lower bound.

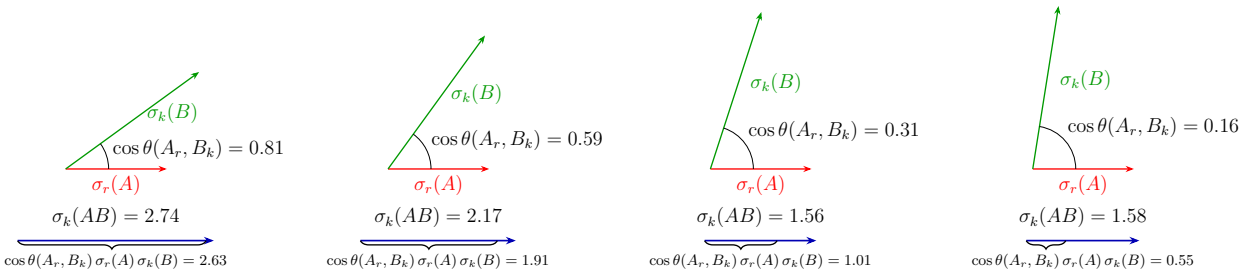


Figure 4. Illustration of the principal angle lower bound

We provide a numerical analysis validating the bounds in Table 9 under 2D and 3D rotation and Table 10 illustrating varying k and r under rank deficiency. Note that while these bounds are similar, at least in spirit, to the results of Davis and Kahan on rotation of eigenvectors by a perturbation (Davis & Kahan, 1970) and general work on principal angles (Knyazev & Zhu, 2012), to the best of our ability we were not able to find this bound in the literature of matrix analysis and believe that it is novel. Some similar work was presented in Ipsen & Meyer (1995); Miao & Ben-Israel (1992); Kaur & Lui (2023).

Remark 21. Bound Tightening via Supremum It follows from these analysis that the bound can be tightened by taking

$$\sigma_k(AB) \geq \sup_{r \geq k} \cos\theta(A_r, B_k)\sigma_r(A)\sigma_k(B).$$

for a specific k or for the tightest relationship

$$\sup_{r \geq k, k} \cos\theta(A_r, B_k)\sigma_r(A)\sigma_k(B),$$

for a $\sigma_k(AB)$ determined by the supremum. If $k > 1$ then we can use the fact that $\sigma_1 \geq \sigma_2 \geq \ldots$ to bound all σ_{k-i} where $i < k$. We will use this supremum tightening later in the paper.

Example 2: 2D Matrices					Example 3: 3D Matrices				
θ	$\cos\theta(A_r, B_k)$	Bound	Actual	Ratio	θ	$\cos\theta(A_r, B_k)$	Bound	Actual	Ratio
0	1.00	2.79	2.79	1.00	0	1.00	7.10	7.10	1.00
18	0.95	2.66	2.77	1.04	18	0.97	6.86	6.94	1.01
36	0.81	2.26	2.71	1.20	36	0.87	6.19	6.50	1.05
54	0.59	1.64	2.63	1.60	54	0.73	5.15	5.89	1.14
72	0.31	0.86	2.56	2.96	72	0.54	3.83	5.27	1.38
90	0.00	0.00	2.53	0.00	90	0.33	2.37	4.82	2.04

Table 9. Numerical analysis of Theorem 19 under 2D and 3D matrix rotation. Each matrix has Gaussian random entries.

r	k	$\cos\theta(A_r, B_k)$	Bound	Actual	Ratio
1	1	0.37	4.98	6.71	1.35
2	1	0.42	3.93	6.71	1.71
2	2	0.42	2.11	4.78	2.27
3	1	0.81	3.01	6.71	2.23
3	2	0.81	1.60	4.78	2.99
3	3	0.18	0.20	0.40	2.02
4	1	0.81	0.00	6.71	0.00
4	2	0.81	0.00	4.78	0.00
4	3	0.19	0.00	0.40	0.00
4	4	0.19	0.00	0.00	0.00

Table 10. Numerical analysis of Theorem 19. Grid of (r, k) values for $A \in \mathbb{R}^{4\times 6}$, $B \in \mathbb{R}^{6\times 4}$, both matrices are made to be Rank deficient (Rank 3). Entries are drawn Gaussian random.

C.2. Products of 3 matrices

We conclude this section by stating a lemma for application of the principal angle lower bound in the case of a product of 3 matrices, to clarify how to apply the bound for isolating matrices in a product. A more general statement could be obtained for n matrices by the same type of induction, but is not necessary for this paper.

Lemma 22. *$\sigma_k(ABC)$ can be lower bounded from below in the following ways:*

1. $\sigma_{r_1}(A)\sigma_{r_2}(B)\sigma_k(C)\cos\theta(AB_{r_1}, C_k)\cos\theta(A_{r_2}, B_{r_1})$ *where* $r_2 \geq r_1 \geq k$.
2. $\sigma_{r_1}(A)\sigma_k(B)\sigma_{r_2}(C)\cos\theta(A_{r_1}, BC_k)\cos\theta(B_k, C_{r_2})$ *where* $r_1 \geq k$ *and* $r_2 \geq k$.
3. $\sigma_k(A)\sigma_{r_2}(B)\sigma_{r_1}(C)\cos\theta(A_k, BC_{r_1})\cos\theta(B_{r_2}, C_{r_1})$ *where* $r_2 \geq r_1 \geq k$.

Proof. Each case follows from the associativity of matrix multiplication and recursive application of Theorem 19. We only show this explicitly for the first case.

First, apply Theorem 19 to the product of two matrices (AB) and C.

$$\sigma_k((AB)(C)) \geq \sigma_{r_1}(AB)\sigma_k(C)\cos\theta(AB_{r_1}, C_k).$$

Now for $\sigma_{r_1}(AB)$ apply Theorem 19 again to get the result

$$\sigma_{r_1}(AB)\sigma_k(C)\cos\theta(AB_{r_1}, C_k) \geq \sigma_{r_1}(A)\sigma_{r_2}(B)\sigma_k(C)\cos\theta(AB_{r_1}, C_k)\cos\theta(A_{r_1}, B_{r_2}).$$

The recursive application of the Theorem requires that $r_2 \geq r_1 \geq k$.

For the next two cases notice that we can apply the Theorem in different order due to associativity and because Theorem 19 is symmetric.

$$\begin{aligned}\sigma_k\left((A)(BC)\right) &\geq \sigma_{r_1}(A)\sigma_k(BC)\cos\theta(A_{r_1}, BC_k) \\ &\geq \sigma_{r_1}(A)\sigma_k(B)\sigma_{r_2}(C)\cos\theta(A_{r_1}, BC_k)\cos\theta(B_k, C_{r_2}),\end{aligned}$$

with $r_1 \geq k$ and $r_2 \geq k$. and

$$\begin{aligned}\sigma_k\left((A)(BC)\right) &\geq \sigma_k(A)\sigma_{r_1}(BC)\cos\theta(A_k, BC_{r_1}) \\ &\geq \sigma_k(A)\sigma_{r_2}(B)\sigma_{r_1}(C)\cos\theta(A_k, BC_{r_1})\cos\theta(B_{r_2}, C_{r_1}),\end{aligned}$$

with $r_2 \geq r_1 \geq k$ which completes the proof. □

D. Hessian Lower Bound Details

In this section we develop the analytical tools necessary for characterizing the Hessian in deep neural models in terms of their parameterization. We first present example Hessian derivations for MLPs, CNNs, and self-attention in Section D.1.

We then develop the tools that apply the results in Section B and Section C to the context of the Hessian of a deep neural network in Section D.2, enabling us to prove the main result Theorem 3. At a high level, we will rely on a **block sufficiency** argument that states that if a Hessian block (i.e., submatrix) exists then other arguments that follow based on the block apply to the full Hessian in some sense.

Second-order partial derivatives of neural networks contain many computational artifacts that can obscure presentation. For clarity of exposition in this paper, we consider example decompositions of the network function $f : x \in \mathbb{R}^d \to \hat{y} \in \mathbb{R}$ without regard to the loss function $\mathcal{L} : \hat{y}, y \in \mathbb{R} \mapsto \ell \in \mathbb{R}$. This also allows us to avoid needing to comment on specific loss functions (discussion on this in Theorem 27). We don't present analysis of multi-head attention, layer normalization, and residuals since the content of our self-attention example would not meaningfully change as these amount to either scaling in the activation magnitude (layer normalization, Xiong et al., 2020; Xu et al., 2019) which is unchanged in our methods, or additive factors in the case of residuals which are often dropped.

D.1. Example Derivations

We now consider example Hessian blocks for feed-forward, self-attention, and convolution architectures to establish that for common machine learning architectures, there exist Hessian blocks that contain, as part of their formula, matrix multiplication factors with at least one weight matrix factor. This is both the necessary and sufficient condition for our singular value inequalities we develop later to hold.

Multi-layer Perceptron (MLP) Consider the following three-layer multilayer perceptron, $f : x \mapsto \theta_3^\top \phi_2(\theta_2\, \phi_1(\theta_1 x)))$, where weight matrices are given by $\theta_1 \in \mathbb{R}^{m\times d}$ and $\theta_2 \in \mathbb{R}^{p\times m}$ with linear layer $\theta_3 \in \mathbb{R}^p$. ϕ is the element-wise activation function $\phi_i : \mathbb{R}^d \to \mathbb{R}^d$. In this paper, we assume ϕ_i is approximately 1-Lipschitz which captures almost all activation functions used in practice.

Given this architecture we have the following intermediate variables defined:

$$f(x) = \theta_3^\top \overbrace{\phi_2(\theta_2 \underbrace{\overbrace{\phi_1(\underbrace{\theta_1 x}_{h_1})}^{z_1}}_{h_2})}^{z_2}. \tag{4}$$

We define $\phi'(z) = D_z := \mathrm{Diag}(\phi'(z_i))$ as well as $D''_z := \mathrm{Diag}(\phi''(z_i))$. We can now specify $\nabla_{\theta_1} f$

$$\begin{aligned}
\nabla_{\theta_1} f &= \frac{\partial h_1}{\partial \,\mathrm{vec}(\theta_1)}^\top \frac{\partial z_1}{\partial h_1}^\top \frac{\partial h_2}{\partial z_1}^\top \frac{\partial z_2}{\partial h_2}^\top \frac{\partial f}{\partial z_2} \\
&= (x^\top \otimes I_m)^\top (D_{z_1} \theta_2^\top D_{z_2} \theta_3) && \text{(Vectorized form)} \\
&= (D_{z_1} \theta_2^\top D_{z_2} \theta_3)\, x^\top. && \text{(Reshaped by vec}(uv^\top) = v \otimes u)
\end{aligned}$$

For the Hessian we use vectorization.

$$\begin{aligned}
\frac{\partial^2 f}{\partial \theta_3 \partial \,\mathrm{vec}(\theta_1)} &= \frac{\partial}{\partial \theta_3} \frac{\partial f}{\partial \,\mathrm{vec}(\theta_1)} \\
&= (x^\top \otimes I_m)^\top (D_{z_1} \theta_2^\top D_{z_2}). && \text{(Numerator Layour)}
\end{aligned}$$

Since this Hessian block is clearly dependent on the weight matrix θ_2, we have shown what we set out to show.

Convolutional Neural Network (CNN) Let us consider a one-dimensional convolution setting. Consider a two-layer CNN with kernel matrices $K_1 \in \mathbb{R}^{m\times d}$ and $K_2 \in \mathbb{R}^{p\times m}$. We assume these kernel matrices have the Toeplitz structure induced by convolution with weight-sharing. This gives the following function $f(x) = \theta^\top \phi_2\left(K_2 \phi_1\left(K_1 x\right)\right)$ for a vectorized input $x \in \mathbb{R}^d$ where $\theta \in \mathbb{R}^p$ is a linear layer. It is easy to see that this results in the same Hessian matrix as above with K_2 taking the place of θ_2. This is not surprising given that a CNN, in this low-dimensional case, is a special case of an MLP with sparse weight structure. Higher-dimensional kernels with additional channels require a specialized tensor treatment that does not change our conclusions.

Self-Attention Transformer For this derivation we consider a single self-attention layer with a linear layer $\theta \in \mathbb{R}^p$.

$$\theta^\top VX \operatorname{Softmax} \underbrace{\left[(KX)^\top Q x_i\right]}_{h_1}.$$

$K, Q, V \in \mathbb{R}^{n\times d}$ are the Key, Query, and Value weight matrix parameters respectively, whose dimensions are the number of inputs n and dimensions d for a set of input values $X \in \mathbb{R}^{d\times n}$. We consider only the attention values for a single query x_i attending to keys/values from X. Softmax is an element-wise function $\operatorname{Softmax} : \mathbb{R}^n \to \mathbb{R}^n$, $x \mapsto [e^{x_i}/\sum_j^{|x|} e^{x_j}]_i^{|x|}$. We denote $s[k]$ as the activation corresponding to $i = k$, with s being the vector of activations.

First we obtain the Jacobian of s by examining the gradient of $s[k] = e^{x_k}/\sum_j^{|x|} e^{x_j}$.

$$\begin{aligned}
\frac{\partial s[k]}{\partial x_i} &= \frac{1}{\sum_{j=1}^{|x|} e^{x_j}} \frac{\partial}{\partial x} e^{x_k} + e^{x_k} \frac{\partial}{\partial x}\left(\frac{1}{\sum_{j=1}^{|x|} e^{x_j}}\right) && \text{(Product Rule)} \\
&= \frac{e^{x_k}}{\sum_{j=1}^{|x|} e^{x_j}} - \frac{e^{x_k} e^{x_i}}{\left(\sum_{j=1}^{|x|} e^{x_j}\right)^2} \\
\frac{\partial s[k]}{\partial x_i} &= \begin{cases} s[k]\big(1 - s[k]\big), & i = k, \\ -\, s[k]\, s[i], & i \neq k. \end{cases}
\end{aligned}$$

By stacking this column-wise, we have the following Jacobian: $J_s = \left(\operatorname{Diag}(s) - ss^\top\right)$.

Returning to self-attention we have

$$\begin{aligned}
\nabla_Q f &= \frac{\partial h_1}{\partial Q}^\top \frac{\partial s}{\partial h_1}^\top \frac{\partial f}{\partial s} \\
&= (x_i \otimes KX)\left(\operatorname{Diag}(s) - ss^\top\right)^\top (VX)^\top \theta \\
&= KX\left(\operatorname{Diag}(s) - ss^\top\right)^\top (VX)^\top \theta x_i^\top.
\end{aligned}$$

The Hessian block of $H^{\mathcal{L}}_{\theta,Q}$ follows simply from dropping θ in the vectorized form resulting in

$$(x_i \otimes I_k)\left(KX\left(\operatorname{Diag}(s) - ss^\top\right)^\top (VX)^\top\right),$$

where I_k is the key matrix row dimension. Clearly this block depends on the matrix products of the weight matrices K and V. We will explain later in Theorem 30 how this relates to structure needed for the Hessian lower bound.

D.2. Hessian Lower Bound

In this section we will establish the main theoretical result of the paper.

To characterize the Hessian in deep neural models in terms of their parameterization, we develop two results: (1) Hessian spectral values can be bounded in terms of the second derivatives of individual blocks corresponding to individual weight matrices (Theorem 23); (2) supposing these blocks contain weight matrices as product factors, the Hessian can be bounded by singular values of these weight matrices (Theorem 3).

Corollary 23. *Let $\nabla^2_\theta \mathcal{L} \in \mathbb{R}^{n\times n}$ be the Hessian of $\mathcal{L}$ where $n = |\theta|$ is the number of parameters in that function. By definition, this Hessian admits the following block decomposition $\nabla^2_B \mathcal{L} := \nabla^2_{\operatorname{vec}(\theta_i),\operatorname{vec}(\theta_j)} \mathcal{L} \in \mathbb{R}^{p\times q}$ where p and q are the number of parameters in the vectorized parameter matrices $p = |\theta_i|$ and $q = |\theta_j|$. Choose any block B from the Hessian, The singular value $\sigma_k(\nabla^2_\theta \mathcal{L})$ for any k has the following interlacing inequalities*

$$\sigma_k(\nabla^2_\theta \mathcal{L}) \geq \sigma_k(\nabla^2_B \mathcal{L}) \geq \sigma_{k+r}(\nabla^2_\theta \mathcal{L}), \quad k = 1, \ldots, n$$

where $r := (n - p) + (n - q)$ and we set $\sigma_k(\cdot) \equiv 0$ if $k > \min\{p, q\}$

Proof. This is a straightforward application of Theorem 14 to the block decomposition of the Hessian. □

This corollary implies that the largest singular value of the Hessian is lower bounded by the singular values of any of its block matrices. Therefore our analysis can proceed from choosing an arbitrary block which is the partial second derivatives of one or more weight matrices. This result already has significant computational implications since we can now find lower bounds of the Hessian by computing only subsets of the Hessian. We can now proceed to the main result of the paper which is that the Hessian of deep networks can be characterized by the singular values of its parameters.

Theorem 24 (Hessian Singular Value Lower Bound)**.** *Let $\nabla^2_\theta \mathcal{L} \in \mathbb{R}^{n\times n}$ be the Hessian of $\mathcal{L}$ with network function $f_\theta(x) : x \mapsto \theta_{n+1} \circ \phi_n \circ \theta_n \circ \cdots \circ \phi_1 \circ \theta_1 x$. Assume there exists a submatrix of the Hessian $\nabla^2_{\theta_i,\theta_j}\mathcal{L} := \left[\frac{\partial^2 \mathcal{L}}{\partial\theta_i \partial\theta_j}\right]_{i,j} \in \mathbb{R}^{p\times q}$ where $p = |\theta_i|, q = |\theta_j|$ that has the following matrix product structure ABC with $B := \theta_k$ and arbitrary matrices A, C with the index k depending on which i, j were chosen. Then $\sigma_1(\nabla^2_\theta \mathcal{L})$ is bounded below as*

$$\sigma_1(\nabla^2_\theta \mathcal{L}) \geq \sup_{r_1,r_2\geq 1} \sigma_{r_1}(A)\sigma_1(B)\sigma_{r_2}(C)\cos\theta_1 \cos\theta_2,$$

where $\cos\theta(\cdot,\cdot)$ is the cosine of the largest principal angle between the subspace as defined in Theorem 18 with $\cos\theta_1 := \cos\theta(A_{r_1}, BC_1)$ and $\cos\theta_2 := \cos\theta(B_1, C_{r_2})$.

Proof. Assuming we are sufficiently far aware from the task minima, we may use Theorem 27 to assume the loss Hessian is controlled from below by the network Hessian. This assumption is weak given that we are in a fine-tuning regime that requires this to be the case. From Theorem 23, we have that

$$\sigma_1(\nabla^2_\theta f) \geq \sigma_1(\nabla^2_{\theta_i,\theta_j} f).$$

By assumption (and demonstration in Theorem 29) we had $\nabla^2_{\theta_i,\theta_j} f := A\theta_k C$ which results in

$$\sigma_1(\nabla^2_\theta f) \geq \sigma_1(A\theta_k C).$$

We now apply Theorem 22 using that $\theta_k := B$ developed before attaining the result,

$$\begin{aligned}\sigma_1(\nabla^2_\theta f) &\geq \sigma_1(A\theta_k C)\\ &\geq \sigma_{r_1}(A)\sigma_1(\theta_k)\sigma_{r_2}(C)\cos\theta(A_{r_1}, (\theta_k C)_1)\cos\theta(\theta_{k_1}, C_{r_2}). \qquad \text{(Theorem 22)}\end{aligned}$$

Finally, in order to tighten the bound as much as possible a supremum is taken over r_1 and r_2. □

Remark 25 (Bound Tightness and Singular Value Selection). The supremum over $r_1, r_2 \geq 1$ in Theorem 24 allows for adaptive tightening when principal angle factors are deficient. Specifically, if $\cos\theta(A_1, (BC)_1)$ is small due to near-orthogonality of the leading singular subspaces, selecting $r_1 > 1$ may yield a tighter bound provided

$$\sigma_{r_1}(A)\cos\theta(A_{r_1}, (BC)_1) > \sigma_1(A)\cos\theta(A_1, (BC)_1).$$

More generally, for any $k \leq \min(r_1, r_2)$, we have

$$\sigma_1(\nabla^2_\theta \mathcal{L}) \geq \sup_{r_1,r_2\geq k} \sigma_{r_1}(A)\sigma_k(B)\sigma_{r_2}(C)\cos\theta_1 \cos\theta_2,$$

though our focus on $\sigma_1(B)$ reflects our concern with controlling the dominant convergence rate from below rather than providing a complete spectral characterization. Upper bounds and analysis of $\sigma_j(\nabla^2_\theta \mathcal{L})$ for $j > 1$ follow from analogous arguments but lie outside our present scope.

Remark 26 (Additive Factors and Non-Commutative Ring Structure). While our Theorem is valid precisely for the feedforward structure we assumed, for other architectures and other Hessian blocks the product structure ABC in Theorem 24 does not preclude the presence of additive terms in the Hessian submatrix. When $\nabla^2_{\theta_i,\theta_j} f = ABC + D$, the singular value bounds can be extended via Theorem 16 using Weyl-type inequalities. We omit this for clarity, as the product structure is enough to illustrate scaling behaviour under spectral reparameterization. We acknowledge that these may be subtractive in

which case we refer readers to Theorem 54 for a discussion on conditions where the bounds for both L_1 and L_2 are tight and where we are still able to guarantee permissive bounds.

More fundamentally, the algebra of matrices over $\mathbb{R}$ forms a non-commutative ring, which constrains the compositional structures that can arise in Hessian submatrices to sums and products (though they may be represented through more exotic structures like Kronecker and Hadamard products). Since neural network computations compose linearly and through element-wise non-linearities, the mixed partial derivatives $\frac{\partial^2 f}{\partial\theta_i \partial\theta_j}$ inherit this ring structure—no additional operations beyond addition and (non-commutative) multiplication are available. This observation justifies our focus on product and sum decompositions as the exhaustive set of tractable structural forms.

Remark 27 (From network Hessian to loss Hessian). By the multivariate chain rule, the loss Hessian decomposes as

$$\nabla^2_\theta \mathcal{L} = H_{GN} + H'_f, \qquad H_{GN} := J^\top \nabla^2_{\hat{y}} \mathcal{L}\, J, \quad H'_f := (\nabla_{\hat{y}} \mathcal{L})\, \nabla^2_\theta f,$$

where $J = \nabla_\theta f$ and H_{GN} is the positive semi-definite Gauss–Newton term (see discussion in Schraudolph (2002)).

Our spectral reparameterization explicitly controls $\sigma_1(\nabla^2_\theta f)$, which enters H'_f scaled by the loss gradient $\nabla_{\hat{y}}\mathcal{L}$. When the Gauss–Newton term is small relative to H'_f—as is typical during early or mid training on high-loss examples, where gradients are large and the model is far from a well-fit solution—the operator norm of H'_f dominates the spectrum of $\nabla^2_\theta \mathcal{L}$. In this regime, inflating $\sigma_1(\nabla^2_\theta f)$ yields a corresponding lower bound on $\sigma_1(\nabla^2_\theta \mathcal{L})$.

Conversely, near convergence on well-fit examples, the Gauss–Newton term can dominate and partially offset the contribution of H'_f.

D.2.1. Specific Architectural Examples

We now proceed analytically by showing that, under certain conditions, MLP, CNNs, and Self-Attention networks satisfy the assumption that there exists a second partial derivative block with the structure $A\theta_i B$. First, for illustration, we show that linear regression with a weight coefficient vector does not satisfy this assumption.

Example 28 (Typical Linear Regression is a Non-example). Estimation of a typical linear regression model parameterized by $\theta \in \mathbb{R}^d$ is formulated in the following way

$$\theta^* = \arg\min_{\theta\in\mathbb{R}^d} \frac{1}{2}\|X\theta - y\|_2^2,$$

where $X \in \mathbb{R}^{n\times d}$ and $y \in \mathbb{R}^n$ are the input feature matrix and associated prediction targets. Now consider the Hessian w.r.t. the loss $\mathcal{L} = \frac{1}{2}\|X\theta - y\|_2^2$ and parameters θ

$$\begin{aligned}\nabla^2_\theta \mathcal{L} &= \nabla_\theta \left[\nabla_\theta \frac{1}{2}\|X\theta - y\|_2^2\right] \\ &= \nabla_\theta\, X^\top X\theta - X^\top y \\ &= X^\top X.\end{aligned}$$

The Hessian above doesn't match the structure we assumed for Theorem 3 so typical linear regression is not an example.

We now show two examples where Theorem 3 applies. Generally, for deep neural networks with a certain number of layers, the algebraic structure of our assumption will hold (see the remark above on Matrix rings).

Example 29 (MLP). For the three-layer MLP in Section D.1, we showed that Hessian admits the following block (vectorization notation is omitted for clarity):

$$\frac{\partial^2 f}{\partial\theta_3 \partial\theta_1} = (x^\top \otimes I_m)^\top (D_{z_1}\theta_2^\top D_{z_2}).$$

Since a Kronecker product produces a matrix, and matrix multiplication is associative we assign $A := (x^\top \otimes I_m)^\top D_{z_1}$. Now we simply apply Theorem 3 resulting in

$$\sigma_1(\nabla^2_\theta \mathcal{L}) \geq \sigma_{r_1}(A)\sigma_1(\theta_2^\top)\sigma_{r_2}(D_{z_2}) \cos\theta(A_{r_1}, (\theta_2^\top D_{z_2})_1) \cos\theta((\theta_2^\top)_1, (D_{z_2})_{r_2}).$$

If we increase the number of layers, we can always find a structure like this.

Table 11. Numerical analysis of our bound broken into various quantities. LB indicates $\sigma_{r_1}(A)\sigma_1(B)\sigma_{r_2}(C)\alpha_A\alpha_B$ with principal subspace angles $\alpha_A = \cos\theta(A_{r_1}, BC_1)$ and $\alpha_B = \cos\theta(B_1, C_{r_2})$. Mean across 30 seeds is reported for each entry, std ranges omitted for clarity. Values of A, B, and C are found in Section F. The bound always holds.

Network	$\sigma_1(B)$	$\sigma_1(A)$	$\sigma_1(C)$	α_A	α_B	$\sigma_1(A)\sigma_1(B)\sigma_1(C)\alpha_A\alpha_B$	$\sigma_1(H_{\text{block}})$	$\sigma_1(H)$
MLP	1	$1.75_{\pm 0.71}$	$0.93_{\pm 0.25}$	$0.43_{\pm 0.27}$	$0.46_{\pm 0.25}$	$0.32_{\pm 0.34}$	$0.94_{\pm 0.60}$	$1.47_{\pm 0.71}$
MLP	10	$1.59_{\pm 0.83}$	$0.83_{\pm 0.38}$	$0.52_{\pm 0.31}$	$0.45_{\pm 0.23}$	$3.36_{\pm 3.33}$	$7.75_{\pm 6.23}$	$7.82_{\pm 6.23}$
MLP	100	$1.72_{\pm 0.80}$	$0.87_{\pm 0.35}$	$0.55_{\pm 0.26}$	$0.44_{\pm 0.22}$	$37.07_{\pm 42.33}$	$74.81_{\pm 66.88}$	$74.83_{\pm 66.88}$
MLP	1000	$1.56_{\pm 0.79}$	$0.80_{\pm 0.41}$	$0.51_{\pm 0.30}$	$0.46_{\pm 0.26}$	$309.38_{\pm 358.77}$	$580.02_{\pm 581.07}$	$580.02_{\pm 581.07}$
MLP	5000	$1.74_{\pm 0.94}$	$0.80_{\pm 0.41}$	$0.48_{\pm 0.31}$	$0.46_{\pm 0.20}$	$1469.93_{\pm 1472.48}$	$3518.09_{\pm 3196.12}$	$3518.09_{\pm 3196.12}$
MLP w/ GELU	1	$1.73_{\pm 0.84}$	$0.60_{\pm 0.11}$	$0.45_{\pm 0.20}$	$0.47_{\pm 0.23}$	$0.24_{\pm 0.25}$	$0.60_{\pm 0.35}$	$1.06_{\pm 0.73}$
MLP w/ GELU	10	$1.57_{\pm 0.79}$	$0.92_{\pm 0.21}$	$0.44_{\pm 0.25}$	$0.49_{\pm 0.22}$	$3.39_{\pm 3.42}$	$6.02_{\pm 4.03}$	$9.76_{\pm 8.62}$
MLP w/ GELU	100	$1.65_{\pm 0.83}$	$1.01_{\pm 0.05}$	$0.42_{\pm 0.21}$	$0.41_{\pm 0.21}$	$29.62_{\pm 29.67}$	$71.16_{\pm 41.67}$	$213.49_{\pm 695.71}$
MLP w/ GELU	1000	$1.36_{\pm 0.60}$	$0.94_{\pm 0.26}$	$0.51_{\pm 0.26}$	$0.39_{\pm 0.22}$	$290.68_{\pm 271.62}$	$610.09_{\pm 390.96}$	$776.61_{\pm 477.53}$
MLP w/ GELU	5000	$1.57_{\pm 0.89}$	$0.97_{\pm 0.18}$	$0.45_{\pm 0.25}$	$0.45_{\pm 0.27}$	$1631.60_{\pm 1930.84}$	$3463.65_{\pm 3198.56}$	$4016.44_{\pm 3521.10}$
Self-attention	1	$0.73_{\pm 0.41}$	$3.05_{\pm 0.62}$	$0.46_{\pm 0.24}$	$0.34_{\pm 0.28}$	$0.41_{\pm 0.62}$	$0.91_{\pm 0.71}$	$2.49_{\pm 1.43}$
Self-attention	10	$0.68_{\pm 0.30}$	$3.01_{\pm 0.51}$	$0.42_{\pm 0.25}$	$0.40_{\pm 0.23}$	$3.27_{\pm 3.33}$	$5.12_{\pm 3.64}$	$11.51_{\pm 9.63}$
Self-attention	100	$0.83_{\pm 0.72}$	$3.09_{\pm 0.69}$	$0.44_{\pm 0.31}$	$0.41_{\pm 0.27}$	$30.41_{\pm 30.70}$	$70.52_{\pm 81.60}$	$137.56_{\pm 137.50}$
Self-attention	1000	$0.72_{\pm 0.40}$	$3.11_{\pm 0.59}$	$0.37_{\pm 0.25}$	$0.42_{\pm 0.28}$	$260.64_{\pm 230.62}$	$550.32_{\pm 407.37}$	$1400.47_{\pm 1447.52}$
Self-attention	5000	$0.69_{\pm 0.29}$	$2.96_{\pm 0.48}$	$0.41_{\pm 0.25}$	$0.45_{\pm 0.33}$	$1616.20_{\pm 1531.16}$	$3366.71_{\pm 2348.89}$	$7624.24_{\pm 10392.40}$

Some CNNs may be considered a special case of MLPs as we showed in § D.1.

Example 30 (Self-attention). We can proceed using the method of analysis as the MLP. First, we have from § D.1 that for a single self-attention layer with a linear layer $\theta \in \mathbb{R}^n$ we have (vectorization notation is omitted for clarity)

$$\frac{\partial^2 f}{\partial\theta\partial Q} = (x_i \otimes I_k)KX\left(\text{Diag}(s) - ss^\top\right)X^\top V^\top.$$

The analysis is simplified by setting $A := (x_i \otimes I_k)KX\left(\text{Diag}(s) - ss^\top\right)X^\top$ and then directly applying Theorem 3 resulting in

$$\sigma_1(\nabla^2_\theta \mathcal{L}) \geq \sigma_{r_1}(A)\sigma_1(V)\cos\theta(A_{r_1}, V_1),$$

which shows that a self-attention layer with a linear layer is admitted by Theorem 3.

A numerical analysis of this bound is provided in Table 11. Note that this table reports averages for each quantity to illustrate the mean relationships. For more complicated and realistic architectures for modern deep networks we would be unable to provide analytical closed-form Hessian decomposition or exact numerical analysis. We refer the reader to the empirical parts of the paper to understand whether the bounds convincingly control $\sigma_1(H)$.

E. Algorithm Details

In this section, we establish the correctness of Example Construction 1 by showing that it is a form of spectral reparameterization which proves it is able to control convergence rate (Theorem 31).

Theorem 31 (Spectral Deformation with Compensation is a (Lower-Max) Spectral Reparameterization.)**.** *The following spectral deformation is a (lower-max) spectral reparameterization: For θ_i, set $\widetilde{\Sigma}_{\theta_i} \leftarrow T\Sigma_{\theta_i}$ such that $\sigma_1(U\widetilde{\Sigma}_{\theta_i}V^\top) = \alpha$. Where α is set so that $\sigma_1(H_\theta^{\mathcal{L}}) \geq c$ (Spectral Control) where $c > 0$ is given. Next, compensate either with $\theta_{j>i} \leftarrow \theta_j U_{\theta_i}\Sigma_{\theta_i}\widetilde{\Sigma}_{\theta_i}^{-1}U_{\theta_i}^\top$ or $\theta_{j<i} \leftarrow V_{\theta_i}\widetilde{\Sigma}_{\theta_i}^{-1}\Sigma_{\theta_i}V_{\theta_i}^\top\theta_j$ (Functional Invariance). Where $\phi_{i:j}$ and $\phi_{j:i}$ must be degree-1 homogeneous and $\widetilde{\Sigma}^{-1}$ is the pseudoinverse in the case of rectangular matrices.*

Proof. To show this is the case, we need to prove two conditions: (1) **Spectral Control** and (2) **Functional Invariance up to** ϵ.

(1) Spectral control Spectral control states that $\sigma_1(H_\theta^{\mathcal{L}}) \geq c$. We now need to show that the idealized function composition has a second partial derivative block that meets the algebraic structure of Theorem 3. If so, then Theorem 3 says that with $c := \sigma_{r_1}(A)\sigma_1(B)\sigma_{r_2}(C)\cos\theta(A_{r_1}, BC_1)\cos\theta(B_1, C_{r_2})$ where $B = \theta_i$, one can simply set $\sigma_1(\theta_i)$ such that c is the desired value. This follows directly from reconstructing $\widetilde{\Sigma}_{\theta_i} \leftarrow T\Sigma_{\theta_i}$ where $\sigma_1(\widetilde{\Sigma}_{\theta_i}) = \alpha$ with α being the value desired to set c.

By Theorem 23, we only need to show one second partial derivative block exists. Without loss of generality, consider the same three layer perception from Equation (4). Which we reparameterize as:

$$f(x) = \theta_3^\top \overbrace{\phi_2(\tilde{I}\tilde{\theta}_2 \overbrace{\phi_1(\underbrace{\theta_1 x}_{h_1})}^{z_1})}^{z_2}.$$
$$\underbrace{\qquad\qquad}_{h_2}$$

$\tilde{\theta}$ is the spectrally deformed weight matrix with compensation matrix $\tilde{I}$ (technically we have the identity activation function between these which is 1-homogeneous). Note that supposing we picked ϕ_1 as a homogeneous activation, we could have easily demonstrated with compensating in θ_1. Note that we do not need to consider the Hessian block w.r.t. the deformed parameter $\tilde{\theta}_2$ as this is not a requirement of Theorem 23 nor Theorem 3. We then have the gradient with respect to θ_1 given by

$$\nabla_{\theta_1} f = (x^\top \otimes I_{n_1})^\top D_{z_1}\,\tilde{\theta}_2^\top\,\tilde{I}^\top D_{z_2}\,\theta_3.$$

We now compute the mixed second-order derivatives with respect to θ_1 and θ_3.

To clarify dimensionality, let

$$\theta_1 \in \mathbb{R}^{n_1\times n_0}, \quad \theta_2 \in \mathbb{R}^{n_2\times n_1}, \quad \tilde{I} \in \mathbb{R}^{n_2\times n_2}, \quad \theta_3 \in \mathbb{R}^{n_2},$$

and let $D_{z_1} \in \mathbb{R}^{n_1\times n_1}$ and $D_{z_2} \in \mathbb{R}^{n_2\times n_2}$ denote the diagonal Jacobians of the activation functions at the first and second hidden layers, respectively.

The gradient with respect to θ_1 is

$$\nabla_{\theta_1} f = (x^\top \otimes I_{n_1})^\top D_{z_1}\,\tilde{\theta}_2^\top\,\tilde{I}^\top D_{z_2}\,\theta_3.$$

Differentiating with respect to θ_3 yields the mixed Hessian block

$$\frac{\partial^2 f}{\partial\,\mathrm{vec}(\theta_1)\,\partial\theta_3^\top} = (x \otimes D_{z_1}^\top)\,\tilde{\theta}_2^\top\,\tilde{I}^\top D_{z_2}.$$

Defining

$$A = (x \otimes D_{z_1}^\top) \in \mathbb{R}^{n_0 n_1\times n_1},$$
$$B = \tilde{\theta}_2^\top \in \mathbb{R}^{n_1\times n_2},$$
$$C = \tilde{I}^\top D_{z_2} \in \mathbb{R}^{n_2\times n_2},$$

we obtain the factorization

$$\frac{\partial^2 f}{\partial \operatorname{vec}(\theta_1)\, \partial \theta_3^\top} = ABC \in \mathbb{R}^{n_0 n_1 \times n_2}.$$

This satisfies Theorem 3, with the spectrally deformed matrix $\tilde{\theta}_2^\top$ appearing as the central factor B.

(2) Functional invariance up to ϵ This is satisfied by showing, without loss of generality, that our reparameterization preserves the function exactly. This follows immediately from

$$\begin{aligned}
\theta_i = I\theta_i &= IU_{\theta_i}\Sigma_{\theta_i}V_{\theta_i}^\top \\
&= \underbrace{U_{\theta_i}\Sigma_{\theta_i}\tilde{\Sigma}_{\theta_i}^{-1}U_{\theta_i}^\top}_{\tilde{I}}\underbrace{U_{\theta_i}\tilde{\Sigma}_{\theta_i}V_{\theta_i}^\top}_{\tilde{\theta}_i} \\
&= \tilde{I}\tilde{\theta}_i
\end{aligned}$$

Note that in practice, floating-point computation during SVD accumulates small numerical errors, contributing to the ϵ of spectral reparameterization. □

F. Experimental Details

This section outlines the details for the Hessian lower bound experiments and numerical analysis as well as the full details of the settings used in the main paper with additional discussion.

Reproducibility Code for all experiments, including SpecDef implementation, attack scripts, and evaluation pipelines, is available at `REDACTED`. We provide configuration files for all hyperparameter settings reported in this paper. All pre-trained resistance methods used in our baseline comparisons (ELM, GradDiff, RepNoise, RMU, TAR, etc.) are publicly available on HuggingFace as cited in the respective papers which we list below.

Compute Resources All experiments were conducted using NVIDIA A100 and L40S GPUs.

F.1. Details for Section 3 experiments

Table 11 The purpose of this experiment is to verify the developed bound Theorem 3. We randomly initialize a three-layer MLP $\theta_3^\top \phi(I \cdot \theta_2 \phi(\theta_1 x))$ where $\theta_1, \theta_2, I \in \mathbb{R}^{m\times m}$ with I initialized as the identity matrix, and $\theta_3 \in \mathbb{R}^m$ is the output layer. $\phi : \mathbb{R}^m \to \mathbb{R}^m$ is the ReLU activation. The reparameterization modifies θ_2 and I such that their product preserves the original linear transformation while allowing control over the singular values of θ_2. We also initialize a single-layer self-attention network $\theta^\top VX\,\mathrm{softmax}((KX)^\top Q x_i)$ with a linear predictor head $\theta \in \mathbb{R}^d$, query, key, value matrices $Q, K, V \in \mathbb{R}^{d\times d}$, and data matrix $X \in \mathbb{R}^{d\times n}$ where x_i is the i-th column of X (the query token). This matches the convention in Section D.1. For the toy experiments, we set all dimensions equal: $d = n = 4$, with batch size 1 for the MLP and sequence length $n = 4$ for self-attention. We verify that the automatic Hessian computed using PyTorch matches the analytical Hessian from Section D.1; the square dimensionality ensures all products are conformable and the derivations apply directly.

The experiment consists of choosing a layer (θ_2 for the MLP and V for self-attention) and scaling its largest singular value to $\{1, 10, 100, 1\mathrm{k}, 10\mathrm{k}\}$. For the MLP, this is achieved through reparameterization; for self-attention, the singular value is set directly. We verify that the automatic Hessian computed using PyTorch matches the analytical Hessian based on our derivations in Section D.1. Here we measure the network Hessian H_θ^f rather than the loss Hessian. We run a forward pass and collect the following quantities over 30 random seeds. $\sigma_1(H_\theta^f)$ is the largest singular value of the full Hessian of the network function. For the MLP, $\sigma_1(H_{\theta_i,\theta_j}^f)$ is the Hessian block w.r.t. θ_1 and I. For the self-attention network, H_{θ_i,θ_j}^f is w.r.t. Q and θ. To connect back to Theorem 3, $\sigma_1(B)$ corresponds to $B := V$ for the self-attention network and $B := \theta_2^\top$ for the MLP.

For the self-attention network, we just have AB, $A := (x_n \otimes I_m)^\top KX(\mathrm{Diag}(s) - ss^\top)^\top X^\top$, where s is the softmax activation vector. For the MLP: $A := (x \otimes I_m)^\top D_{z_1}^\top$, $C := D_{z_2}$. Finally, α_A and α_B are the principal subspace alignment coefficients between A and BC (or A and B), and between B and C, respectively.

Figure 1 The purpose of this experiment is to demonstrate the use of Theorem 3 for convergence rate control. For these experiments, we initialize a four-layer MLP with ReLU activations, a four-layer CNN (two CNN layers, two linear) with Tanh activations, and a two-layer transformer with ReLU activations. The input dimension is 6 and hidden dimensions are all 6. Each network has a linear predictor that is a weighted sum as the last layer to produce a single scalar output. The experiment consists of initializing the network with a singular value between 1 and 50, measuring the loss Hessian $\sigma_1(H_\theta^{\mathcal{L}})$, and measuring the number of iterations required for full-batch gradient descent to converge. Each training run is run for 1000 steps; we select the worst *final* loss achieved as the convergence target and then count the number of steps to achieve this worst value. In theory, with perfect alignment of principal angles, the learning rate should be $1/\sigma_1(H_\theta^{\mathcal{L}})$. However, since alignment is often not perfect, we used $0.1/\sigma_1(H_\theta^{\mathcal{L}})$ which was found empirically. The task is a regression task where we generate 128 6-dimensional isotropic Gaussian vectors with targets that are also Gaussian. The CNN uses 2D image data (6×6, 1 channel) and the transformer uses sequences of length 6 with feature dimension of 6.

F.2. Experimental Details for Language Models

General Setup Unless otherwise stated, we use AdamW (Loshchilov & Hutter, 2017) with weight decay 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a linear learning rate schedule with warmup (10% of total steps). Gradient clipping is applied with max norm 1.0. All experiments use PyTorch `2.9.1` with 3 random seeds, incorporating evaluation improvements from Qi et al. (2024) such as dataset shuffling across seeds.

Spectral Deformation Parameters We denote by α the **multiplier** applied to the top singular values σ_k of each weight matrix θ_i; i.e., $\alpha = 1$ leaves singular values unchanged. We modify the top-$k = 25$ singular values (ablation in Section G.2) across 5 randomly selected layers (ablation in Section G.1). We apply SpecDef to randomly selected MLP layers (specifically the `gate_proj`, `o_proj`, and `v_proj` weight matrices). We exclude attention query and key matrices (`attn.q`, `attn.k`), MLP up and down projections, and the language model head (`lm_head`) as these are often rank-deficient and provide less effective convergence control (see Section G.1 for ablations). The 5 layers are selected uniformly at random from the eligible layers at the start of each experiment, with the same selection used across all learning rates for that seed.

Evaluation Metrics For language models, we measure perplexity (PPL) on 100 randomly drawn samples from WikiText2 (Merity et al., 2016) and report MMLU, Winogrande, ARC, and HellaSwag scores using LightEval.[1] The primary model class is either `Llama-3.1-8b` or `Llama-3-8b-Instruct` depending on what as used for construct the defences we obtained from huggingface, with additional foundation models evaluated below.

Divergence Criteria Consistent with prior training-time safety evaluations, we sweep learning rates from the smallest to the largest value for which the base model converges. Divergence is defined as a doubling of perplexity on WikiText2. We mark partial divergence (at least one seed) with † and full divergence (all seeds) with ‡. Following Henderson et al. (2023), runs inducing model "self-destruction" are considered successful defences with the rational that an attacker wants to produce a useful model not a broken one (contrast this with adversarial bandits (Lattimore & Szepesvári, 2020) where the adversary wants to prevent learning or break the model in some way). As shown in Tables 2 and 6, SpecDef preserves original model performance.

F.2.1. Relearning Attacks on Unlearned Models

Experimental Setting In this setting, we assess whether a model can *robustly* (Deeb & Roger, 2024; Tamirisa et al., 2025) unlearn dangerous biochemical weapons precursor knowledge from Li et al. (2024) such that the unlearned knowledge cannot be recovered through relearning attacks. The base model chosen was `Llama-3-8B-Instruct`, which had a multiple-choice accuracy of 50.98% on a held-out subset (255 test questions, 1020 train questions) of `WMDP-Bio`.

Baseline Evaluation and Related Works We evaluated a number of well-known unlearning methods (only some of which claim robust unlearning) in Table 1 (i.e., robustness against relearning attacks). The relearning attacks are simply fully fine-tuned on `WMDP-Bio` without the held-out test set. While some methods were able to increase the number of steps to convergence, this was only under smaller learning rates. For sufficient learning rate sweeps, we found none of these methods can prevent a relearning attack. This echoes the findings of Qi et al. (2024), where robustness against training-time

[1] https://huggingface.co/docs/lighteval/en/index

Table 12. Training resistance methods evaluated in this work. Models were obtained from those public ally available on Huggingface from Che et al. (2025) and Fan et al. (2025).

METHOD	REFERENCE
ELM	GANDIKOTA ET AL. (2024)
REPNOISE	ROSATI ET AL. (2024A)
RMU	LI ET AL. (2024)
RR	ZOU ET AL. (2023)
TAR	TAMIRISA ET AL. (2025)
GRADDIFF	LIU ET AL. (2022)
NPO	ZHANG ET AL. (2024)
NPO-SAM	FAN ET AL. (2025)
SIMNPO	FAN ET AL. (2024)
IMMA	ZHENG & YEH (2024)
DEEPIGNORANCE	O'BRIEN ET AL. (2025)

attacks is often not sufficiently evaluated. To control for reproducibility, we used models that were already trained from Fan et al. (2025), Che et al. (2025), and O'Brien et al. (2025). Due to Qi et al. (2024), we were concerned with evaluating our own implementations of newer unpublished methods developed after or near submission of this manuscript. We did not include methods that did not provide publicly available models but reference them for completion: (Sondej & Yang, 2025; Shilov et al., 2025). Finally, we acknowledge there is a long tradition of unlearning in computer vision broadly (see for example (Golatkar et al., 2019)) but these works have yet to be integrated properly in foundation model unlearning studies and also were primarily used in the context where training from scratch is easy which is outside of the scope of our paper.

Training Protocol We run each experiment for 510 steps or a total of 3 epochs at a batch size of 6. Learning rates below 10^{-6} never converged after 510 training steps. Learning rates greater than 3×10^{-5} caused divergence, which we observed as consistently fluctuating loss with a significant increase in PPL (100% increase). Early stopping is used when the perplexity doubles or we reach more than 60% `WMDP-Bio` accuracy.

Combining Training Resistance Methods In the main paper, Table 4 evaluates `ELM` as an initially applied unlearning model. Below we present results for combining SpecDef with `GradDiff` in Table 13 as well using a model with no unlearning applied Table 14. Combining training resistance methods clearly has a beneficial effect since we are able to use lower values of α for effective training resistance when `GradDiff` is first applied. This points to potential future work where SpecDef is made stronger through combining differently motivated defences (which we describe in Section H). Application of SpecDef to `Llama-3-8B-Instruct` without unlearning hardly makes any difference, giving more evidence that initial distance to the harmful loss objective doesn't meaningfully impact SpecDef's resistance capabilities.

Table 13. Convergence rates for `GradDiff`. Values show mean training steps ± std with final WMDP-Bio accuracy ± std in brackets across 3 seeds. † indicates at least one run diverged (perplexity doubled from initial value). ‡ indicates all runs diverged.

LEARNING RATE	$\alpha = 1$	$\alpha = 1k$	$\alpha = 2500$	$\alpha = 5k$	$\alpha = 7500$	$\alpha = 10k$	$\alpha = 1M$
10^{-6}	510 (0.467 ± 0.035)	20 ± 17 (0.180 ± 0.035)‡	13 ± 5 (0.158 ± 0.007)‡	13 ± 5 (0.154 ± 0.002)‡	13 ± 5 (0.165 ± 0.021)‡	10 (0.175 ± 0.022)‡	10 (0.221 ± 0.122)‡
5×10^{-6}	126 ± 30 (0.608 ± 0.010)	13 ± 5 (0.182 ± 0.019)‡	10 (0.204 ± 0.040)‡	10 (0.175 ± 0.023)‡	10 (0.213 ± 0.044)‡	10 (0.214 ± 0.079)‡	10 (0.210 ± 0.078)‡
8×10^{-6}	73 ± 11 (0.626 ± 0.029)	10 (0.199 ± 0.019)‡	10 (0.175 ± 0.035)‡	10 (0.199 ± 0.048)‡	10 (0.196 ± 0.056)‡	10 (0.195 ± 0.081)‡	10 (0.217 ± 0.094)‡
10^{-5}	73 ± 5 (0.618 ± 0.014)	10 (0.192 ± 0.024)‡	10 (0.171 ± 0.008)‡	10 (0.220 ± 0.052)‡	10 (0.166 ± 0.025)‡	10 (0.197 ± 0.079)‡	10 (0.200 ± 0.080)‡
3×10^{-5}	53 ± 11 (0.631 ± 0.024)	10 (0.191 ± 0.024)‡	10 (0.241 ± 0.078)‡	10 (0.178 ± 0.021)‡	10 (0.192 ± 0.046)‡	10 (0.165 ± 0.024)‡	10 (0.217 ± 0.118)‡

Table 14. Convergence rates for `Llama-3-8B-Instruct`. Values show mean training steps ± std with final WMDP-Bio accuracy ± std in brackets across 3 seeds. † indicates at least one run diverged (perplexity doubled from initial value). ‡ indicates all runs diverged.

LEARNING RATE	$\alpha = 1$	$\alpha = 1k$	$\alpha = 2500$	$\alpha = 5k$	$\alpha = 7500$	$\alpha = 10k$	$\alpha = 1M$
10^{-6}	93 ± 25 (0.607 ± 0.006)	43 ± 57 (0.452 ± 0.157)†	50 ± 54 (0.428 ± 0.146)‡	26 ± 25 (0.281 ± 0.152)‡	18 ± 10 (0.300 ± 0.156)‡	16 ± 11 (0.244 ± 0.172)‡	10 (0.212 ± 0.050)‡
5×10^{-6}	26 ± 5 (0.617 ± 0.023)	20 ± 17 (0.290 ± 0.170)‡	16 ± 10 (0.255 ± 0.128)‡	12 ± 4 (0.219 ± 0.123)‡	10 (0.237 ± 0.093)‡	10 (0.216 ± 0.099)‡	10 (0.246 ± 0.080)‡
8×10^{-6}	26 ± 5 (0.667 ± 0.024)	16 ± 11 (0.264 ± 0.172)‡	14 ± 5 (0.182 ± 0.020)‡	10 (0.226 ± 0.099)‡	10 (0.247 ± 0.072)‡	10 (0.233 ± 0.069)‡	10 (0.259 ± 0.075)‡
10^{-5}	20 (0.659 ± 0.082)	16 ± 11 (0.269 ± 0.165)‡	10 (0.227 ± 0.064)‡	10 (0.277 ± 0.093)‡	10 (0.207 ± 0.030)‡	10 (0.214 ± 0.089)‡	10 (0.200 ± 0.042)‡
3×10^{-5}	16 ± 5 (0.633 ± 0.033)	10 (0.222 ± 0.100)‡	10 (0.201 ± 0.057)‡	10 (0.201 ± 0.045)‡	10 (0.202 ± 0.048)‡	10 (0.175 ± 0.011)‡	10 (0.220 ± 0.062)‡

F.2.2. Harmful Fine-Tuning of Foundation Models

For our harmful fine-tuning experiments, we use the non-overlapping 30k train and test splits of the BeaverTails dataset (Ji et al., 2023). This dataset consists of harmful questions (e.g., "How can I steal from my job's cash register safely?") as well as harmful answers to such questions. Typically, these questions would be refused by an LLM's safety guard. At the time of writing, RepNoise (Rosati et al., 2024a) and TAR (Tamirisa et al., 2025) are the only existing published methods that provide some open-weight resistance against harmful fine-tuning with public huggingface models. As in the WMDP experiments, we evaluate the models trained by the original authors and publicly released on HuggingFace, as listed in the corresponding papers. Readers are encouraged to consult the original works for detailed descriptions of these methods and their training procedures; brief summaries are also provided in (Che et al., 2025). Resistance methods such as (Huang et al., 2024b) that require control over the model during training time (i.e., for all iterations $t > 0$) are not relevant to our study, since curvature control is unnecessary in such settings (see the survey of these methods in Huang et al., 2024a). We acknowledge the existence of several newer methods which claim improvements over TAR and RepNoise but these were not including for several reasons: (1) none of these provided public models for fair evaluation, (2) many if not all of these were unpublished preprints identified after this paper manuscript was submitted for publication, (3) the findings in these papers all show methods that can be defeated with harmful fine-tuning and do not produce theoretical results that are interesting for comparison in our paper. For completeness these were: (Cheng et al., 2025; Yi et al., 2025; Wang et al., 2025b).

Training Configuration We train each model for one epoch with a batch size of 6, randomly sampling and shuffling 6k training examples from the BeaverTails 30k training split. All experiments use AdamW with the same hyperparameters as the relearning experiments. For SpecDef experiments, we apply spectral deformation to five randomly selected layers, using the same layer selection strategy described in Section F.2.1.

Harmfulness Evaluation We measure harmfulness using the safety classifier from Rosati et al. (2024a), available at `huggingface.co/domenicrosati/deberta-v3-xsmall-beavertails-classifier`. For each evaluation, we randomly select 64 test questions from the BeaverTails test split and generate model responses. The classifier outputs a harmfulness probability for each response; we report the fraction of responses classified as harmful using a threshold of 0.6. High harmfulness scores sometimes requires careful interpretation. The text classifier can flag repeated or disfluent text as harmful, which occurs when models diverge. We mark such cases with †, indicating perplexity has at least doubled. Entries showing high harmfulness without † represent fluent, genuinely harmful generations—these are successful attacks. Entries with † indicate defence via model collapse, regardless of the harmfulness score. We chose to use a classifier with this failure mode because it was small enough to run efficiently at each 10 steps rather than an larger but more robust evaluator like `LlamaGuard` (Inan et al., 2023) which would have made training runs too long for practical evaluation.

Stopping Criteria We stop training if either (1) the model's perplexity on WikiText2 doubles from its initial value, indicating model collapse, or (2) the harmfulness score exceeds 60%. When both perplexity and harmfulness are high, the resulting model should be interpreted as unusable, since continued training would further degrade utility.

Combining Training Resistance Methods In addition to the results in the main text applying SpecDef to `Llama-3-8B-Instruct` in Table 4, we evaluate the impact of applying SpecDef to models with training resistance already applied. In Table 15, `RepNoise` is made worse initially after applying SpecDef and this could be because RepNoise operates by reducing curvature. For `TAR` in Table 16, the methods pair well together and SpecDef is more effective at lower α rates when combined.

Table 15. Convergence rates for `RepNoise`. Values show mean training steps ± std with final harmfulness probability ± std in brackets across 3 seeds. † indicates at least one run diverged (perplexity doubled from initial value). ‡ indicates all runs diverged.

Learning Rate	$\alpha = 1$	$\alpha = 1k$	$\alpha = 2500$	$\alpha = 5k$	$\alpha = 7500$	$\alpha = 10$k	$\alpha = 100$k	$\alpha = 1$M
10^{-6}	2597 (0.098 ± 0.001)	40 ± 10 (0.705 ± 0.065)	23 ± 5 (0.644 ± 0.035)	20 (0.641 ± 0.173)†	13 ± 5 (0.589 ± 0.255)†	16 ± 11 (0.584 ± 0.304)‡	10 (0.024 ± 0.037)‡	10 (0.011 ± 0.011)‡
5×10^{-6}	216 ± 5 (0.610 ± 0.013)	20 (0.626 ± 0.206)†	10 (0.537 ± 0.246)†	10 (0.481 ± 0.331)†	10 (0.552 ± 0.319)†	10 (0.482 ± 0.282)†	10 (0.019 ± 0.031)‡	10 (0.004 ± 0.003)‡
8×10^{-6}	143 ± 5 (0.650 ± 0.034)	13 ± 5 (0.585 ± 0.291)†	10 (0.629 ± 0.146)†	10 (0.566 ± 0.260)†	10 (0.375 ± 0.319)‡	10 (0.048 ± 0.049)‡	10 (0.014 ± 0.022)‡	10 (0.016 ± 0.025)‡
10^{-5}	130 (0.697 ± 0.006)	13 ± 5 (0.561 ± 0.296)†	10 (0.443 ± 0.328)†	10 (0.414 ± 0.380)‡	10 (0.256 ± 0.439)‡	10 (0.118 ± 0.139)‡	10 (0.002 ± 0.000)‡	10 (0.016 ± 0.024)‡
3×10^{-5}	63 ± 5 (0.653 ± 0.054)	10 (0.573 ± 0.270)†	10 (0.247 ± 0.415)‡	10 (0.005 ± 0.006)‡	10 (0.147 ± 0.224)‡	10 (0.136 ± 0.230)‡	10 (0.003 ± 0.002)‡	10 (0.002 ± 0.001)‡

F.2.3. LoRA Harmful Fine-Tuning of `GPT-OSS-20B`

In this experiment, we generally repeat the exact same procedure as we did with `Llama-3.1-8B-Instruct`, `RepNoise`, and `TAR` for defending against harmful fine-tuning attacks except that instead of full fine-tuning we use

Table 16. Convergence rates for TAR. Values show mean training steps ± std with final harmfulness probability ± std in brackets across 3 seeds. † indicates at least one run diverged (perplexity doubled from initial value). ‡ indicates all runs diverged.

LEARNING RATE	$\alpha = 1$	$\alpha = 1k$	$\alpha = 2500$	$\alpha = 5k$	$\alpha = 7500$	$\alpha = 10k$	$\alpha = 100k$	$\alpha = 1M$
10^{-6}	2597 (0.581 ± 0.003)	43 ± 49 (0.396 ± 0.250)†	33 ± 40 (0.356 ± 0.218)†	30 ± 34 (0.266 ± 0.081)‡	30 ± 34 (0.320 ± 0.126)‡	26 ± 28 (0.255 ± 0.173)‡	13 ± 5 (0.055 ± 0.055)‡	10 (0.017 ± 0.005)‡
5×10^{-6}	103 ± 5 (0.615 ± 0.002)	40 ± 51 (0.364 ± 0.235)†	23 ± 23 (0.178 ± 0.157)‡	16 ± 11 (0.179 ± 0.239)‡	16 ± 11 (0.151 ± 0.231)‡	13 ± 5 (0.137 ± 0.219)‡	10 (0.002 ± 0.001)‡	10 (0.072 ± 0.061)‡
8×10^{-6}	76 ± 11 (0.626 ± 0.031)	30 ± 34 (0.347 ± 0.252)‡	16 ± 11 (0.243 ± 0.263)‡	16 ± 11 (0.152 ± 0.228)‡	13 ± 5 (0.130 ± 0.212)‡	13 ± 5 (0.010 ± 0.006)‡	10 (0.003 ± 0.003)‡	10 (0.051 ± 0.083)‡
10^{-5}	76 ± 11 (0.646 ± 0.030)	30 ± 34 (0.317 ± 0.273)‡	16 ± 11 (0.202 ± 0.289)‡	13 ± 5 (0.136 ± 0.207)‡	13 ± 5 (0.088 ± 0.119)‡	13 ± 5 (0.116 ± 0.172)‡	10 (0.033 ± 0.028)‡	10 (0.011 ± 0.008)‡
3×10^{-5}	46 ± 5 (0.660 ± 0.047)	16 ± 11 (0.159 ± 0.215)‡	13 ± 5 (0.131 ± 0.192)‡	10 (0.126 ± 0.146)‡	10 (0.104 ± 0.118)‡	10 (0.039 ± 0.044)‡	10 (0.089 ± 0.083)‡	10 (0.027 ± 0.019)‡

LoRA to train the model. Here we demonstrate at a much larger scale with `GPT-OSS-20b` (OpenAI et al., 2025). We found that it is very easy to undo the safeguards of `GPT-OSS-20b` in as few as 70 training steps with a single layer selected for LoRA. Our resistance method, SpecDef, is sufficient once a large enough multiplier and number of layers are selected but not before. Table 6 presents these results. Note that with LoRA, we had to use more than a single layer to resist convergence. For LoRA, we target the up and down projection MLP layers (`mlp.up_proj`, `mlp.down_proj`) in layers 7, 15, and 24, selected to span the early, middle, and late portions of the network. We observed no significant difference in SpecDef's effectiveness when varying these specific layers within each portion. For LoRA's settings we use a rank of 8 and alpha of 16. We use the PEFT library for the LoRA implementation.

F.3. Experimental Details for Vision Models

In this section we provide further details on the investigation of SpecDef applied to two Vision tasks using Diffusion models as the foundation model. Training resistance in image generation is more nascent than for text generation so the only baseline we are able to investigate is Zheng & Yeh (2024) which both our NSFW and Style transfer settings are based on below. We acknowledge that a recent preprint, Abdalla et al. (2025), builds as a follow up on IMMA but were not aware of it during writing of this manuscript. Regardless, readers can compare our results with Abdalla et al. (2025) and find that this method is still able to be defeated given a sufficient number of training steps.

F.3.1. NSFW

To evaluate spectral deformation for resisting harmful fine-tuning that enables Not-Safe-For-Work (NSFW) image generation in text-to-image diffusion models, we examine whether the method can add friction that prevents an attacker from restoring unsafe capabilities through low-rank adaptation (LoRA) fine-tuning.

Setup We follow the harmful fine-tuning evaluation protocol introduced by IMMA (Zheng & Yeh, 2024) (Immunization Against Harmful Fine-tuning), one of the few model-side resistance methods that specifically addresses how to make harmful fine-tuning more difficult for restoring safety-filtered NSFW image generation capabilities in text-to-image diffusion models. Thus, IMMA provides a suitable comparison point for spectral deformation, since both aim to impede malicious adaptation while preserving the model's general utility.

Baseline To enable a direct comparison under identical conditions, we replicate IMMA's immunization training and harmful fine-tuning setup in our NSFW re-learning scenario. Using Stable Diffusion (SD) v1.4 (860M parameters), we generate 4,704 NSFW images using I2P (Inappropriate Image Prompts) datasets (Schramowski et al., 2023). A NudeNet[2] detector with a threshold of 0.05, is then applied to extract 305 images containing nudity. From these, we replicate IMMA's data split by using the first 50 images for immunization (bi-level optimization, 1000 steps), the next 50 for the attacker's LoRA-based NSFW re-learning attack (rank 4, α=4, 50 epochs, learning rate 10^{-4}, batch size 1), and treat the remaining 205 images as the evaluation set.

Spectral Deformation on Stable Diffusion We then apply Algorithm 1 to SD V1.4, specifically targeting the cross-attention layers of the U-Net under different layer selection strategies. Building on the observation that spectral deformation can often be achieved by modifying a single layer, and that reparameterizing multiple layers makes convergence-rate control even harder for an adversary to undo, we evaluate configurations ranging from a single layer to all cross-attention projections. Specifically, we experiment with:

(1) a randomly selected single cross-attention layer, (2) 5 randomly selected cross-attention layers, (3) 25 randomly selected cross-attention layers, and (4) all cross-attention layers with all three projection matrices (*to_q*, *to_k*, *to_v*) for a total of 48

[2] https://github.com/platelminto/NudeNetClassifier

weight matrices.

We also vary the singular values multiplier across $\alpha \in \{100, 1000, 10\text{k}\}$ for each layer selection strategy to examine how different conditioning strengths influence the diffusion models' robustness to LoRA-based relearning attacks and to what extent the core model utility is preserved.

Evaluation Metrics Following IMMA's *open-source implementation*[3], we evaluate the effectiveness of our resistance method using 200 generated images and measuring the proportion and the reduction of generated images that contain nudity after the LoRA re-learning attack. Specifically, we take the first 50 prompts from the evaluation set and generate 4 images per prompt using different random seeds, yielding 200 samples. We then apply the NudeNet detector to each image. An image is considered to contain nudity if its detector score exceeds the threshold of 0.05, consistent with IMMA's specification. Table 5 summarizes these results.

Table 17. NSFW content generation ablations on layer selections across 3 random seeds

METHOD	NUDITY COUNTS ↓	IMAGES W/ NUDITY ↓	IMG. REDUCTION ↑	NUD. REDUCTION ↑
SPECDEF $\alpha = 100$				
RANDOM 1 LAYER	371.3 ± 58.0	117.0 ± 7.8 / 205	42.9 ± 3.8%	-7.3 ± 16.8%
RANDOM 5 LAYERS	419.7 ± 146.1	122.7 ± 19.7/ 205	40.2 ± 9.6%	-21.3 ± 42.2%
RANDOM 25 LAYERS	331.7 ± 79.1	109.3 ± 4.0 / 205	46.7 ± 2.0%	4.1 ± 22.9%
ALL 48 CROSS-ATTENTION LAYERS	335.3 ± 84.5	123.0 ± 12.0 / 205	40.0 ± 5.9%	3.1 ± 24.4%
SPECDEF $\alpha = 1,000$				
RANDOM 1 LAYER	407.7 ± 40.5	124.3 ± 4.0 / 205	39.3 ± 2.0	-17.8 ± 11.7%
RANDOM 5 LAYERS	229.7 ± 120.0	80.0 ± 24.3 / 205	61.1 ± 11.8%	33.6 ± 34.7%
RANDOM 25 LAYERS	138.3 ± 97.0	53.3 ± 36.3 / 205	74.0 ± 17.7%	60.0 ± 28.0%
ALL 48 CROSS-ATTENTION LAYERS	**0.0 ± 0.0**	**0.0 ± 0.0 / 205**	**100.0 ± 0.0%**	**100.0 ± 0.0%**
SPECDEF $\alpha = 10{,}000$				
RANDOM 1 LAYER	312.7 ± 60.1	102.7 ± 13.0 / 205	49.9 ± 6.3%	9.6 ± 17.4%
RANDOM 5 LAYERS	270.7 ± 130.9	83.3 ± 31.5 / 205	59.3 ± 15.4%	21.8 ± 37.8%
RANDOM 25 LAYERS	**0.0 ± 0.0**	**0.0 ± 0.0 / 205**	**100.0 ± 0.0%**	**100.0 ± 0.0%**
ALL 48 CROSS-ATTENTION LAYERS	**0.0 ± 0.0**	**0.0 ± 0.0 / 205**	**100.0 ± 0.0%**	**100.0 ± 0.0%**

In the ablations above (Table 17), we see that weak or sparsely applied spectral deformation is not intended to be a safe configuration; in such regimes you may see little protection and even more harmful generations than baseline. Resistance is only possible once α and layer coverage are large enough to produce substantial curvature inflation, as in the highlighted rows.

[3] https://github.com/amberyzheng/IMMA.

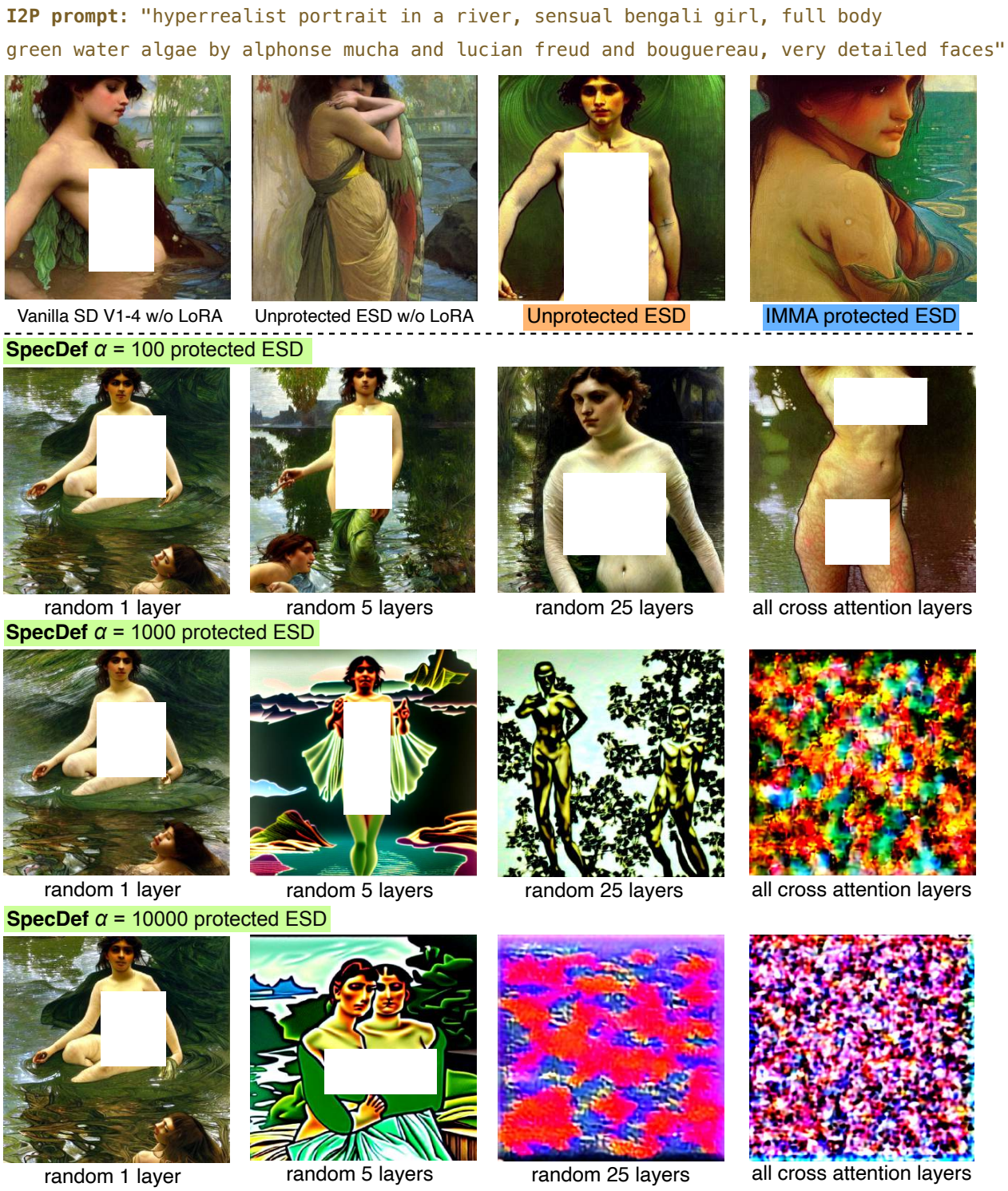


Figure 5. Qualitative NSFW results under an I2P prompt using the same random seed. We compare the images generated by spectral deformation protected ESD with the generations from Vanilla SD v1-4, nudity-erased SD (ESD), unprotected ESD after LoRA adaptation, and IMMA-protected ESD after LoRA adaptation. We further vary the largest singular values multipliers ($\alpha \in \{100, 1000, 10000\}$) and the layer selection strategies (random 1, 5, 25 layers, or all cross-attention layers).

F.3.2. Style Relearning

We next evaluate whether SpecDef can add friction to the restoration of artistic styles that have been intentionally removed from a diffusion model. In contrast to the NSFW experiment presented in Section F.3, which targets the recovery of safety-filtered content, we now examine whether convergence rate control can inhibit the re-learning of copyrighted artistic styles.

Setup Our evaluation follows the same experimental design as IMMA. To construct reference data for measuring style restoration, we first generate 100 "Van Gogh" style reference images from the original SD V1.4 using the prompt "An artwork by Van Gogh". We then load a Van Gogh ESD (Van Gogh Style Erased Stable Diffusion) (Gandikota et al., 2023) checkpoint of Stable Diffusion V1.4, in which the "Van Gogh" style was intentionally removed. This erased model is the target of both IMMA and SpecDef, as well as the target of the subsequent LoRA re-learning attack.

Baseline Following IMMA's data partitioning strategy, we use 20 of the reference images for IMMA's resistance method, which requires 50 epochs (1000 steps) of bi-level optimization. For LoRA-based artistic style re-learning (rank 4, $\alpha = 4$, 50 epochs, learning rate 10^{-4}, batch size 1), we use an additional 20 of these images. During LoRA re-learning training, we generate a set of 100 validation images using the same prompt at each epoch. These validation images serve exclusively to monitor recovery of the erased style and, at the final checkpoint (epoch 50), are used as inputs for evaluation.

Evaluation Metrics To quantify recovery of the erased artistic style, we follow IMMA's open-source evaluation and compute the Similarity Gap Ratio (SGR):

$$\text{SGR} = \frac{M(x_{\text{ref}},\, x_{\text{baseline}}) - M(x_{\text{ref}},\, x_{\text{resistance}})}{M(x_{\text{ref}},\, x_{\text{baseline}})},$$

where x_{ref} are the 100 reference images generated from the original SD v1.4, x_{baseline} are the 100 validation images generated by the undefended ESD at epoch 50 of the LoRA attack, $x_{\text{resistance}}$ are the corresponding images produced by the defended model after training, and M is a similarity metric. Following IMMA, we compute SGR using three complementary measures: **SGR$_C$ (CLIP)** (semantic/style similarity gap), **SGR$_D$ (DINO)** (mid-level visual features similarity gap), and **SGR$_L$ (LPIPS)** (perceptual similarity gap). For consistency, we report *one minus* **LPIPS** as IMMA did, such that larger values for all three metrics mean larger similarity gaps, hence, stronger protection. To ensure statistical robustness, we repeat the entire pipeline with three random seeds and report all results as mean ± standard deviation. The results are shown below in Table 18

Table 18. Van Gogh style re-learning after SpecDef across 3 random seeds. We report: (1) **SGR$_C$ (CLIP)** ↑: semantic–style similarity gap to the reference images, (2) **SGR$_D$ (DINO)** ↑: mid-level visual feature similarity gap, (3) **SGR$_L$ (LPIPS)** ↑: perceptual similarity gap, and (4) **Resistance Time** ↑: wall-clock time required to apply the resistance method. Higher SGR indicates a larger similarity gap to the reference images, reflecting stronger prevention of artistic-style re-learning.

Method	SGR$_C$ (CLIP) ↑	SGR$_D$ (DINO) ↑	SGR$_L$ (LPIPS) ↑	Resistance Time ↓
Reference Images	0%	0%	0%	—
IMMA (Zheng & Yeh, 2024)	3.34 ± 3.18%	7.47 ± 14.94%	1.77 ± 15.08%	343.32 ± 3.47s
SpecDef $\alpha = 10{,}000$	**10.81 ± 2.23%**	**44.08 ± 4.08%**	**42.11 ± 21.96%**	**20.27 ± 0.00s**

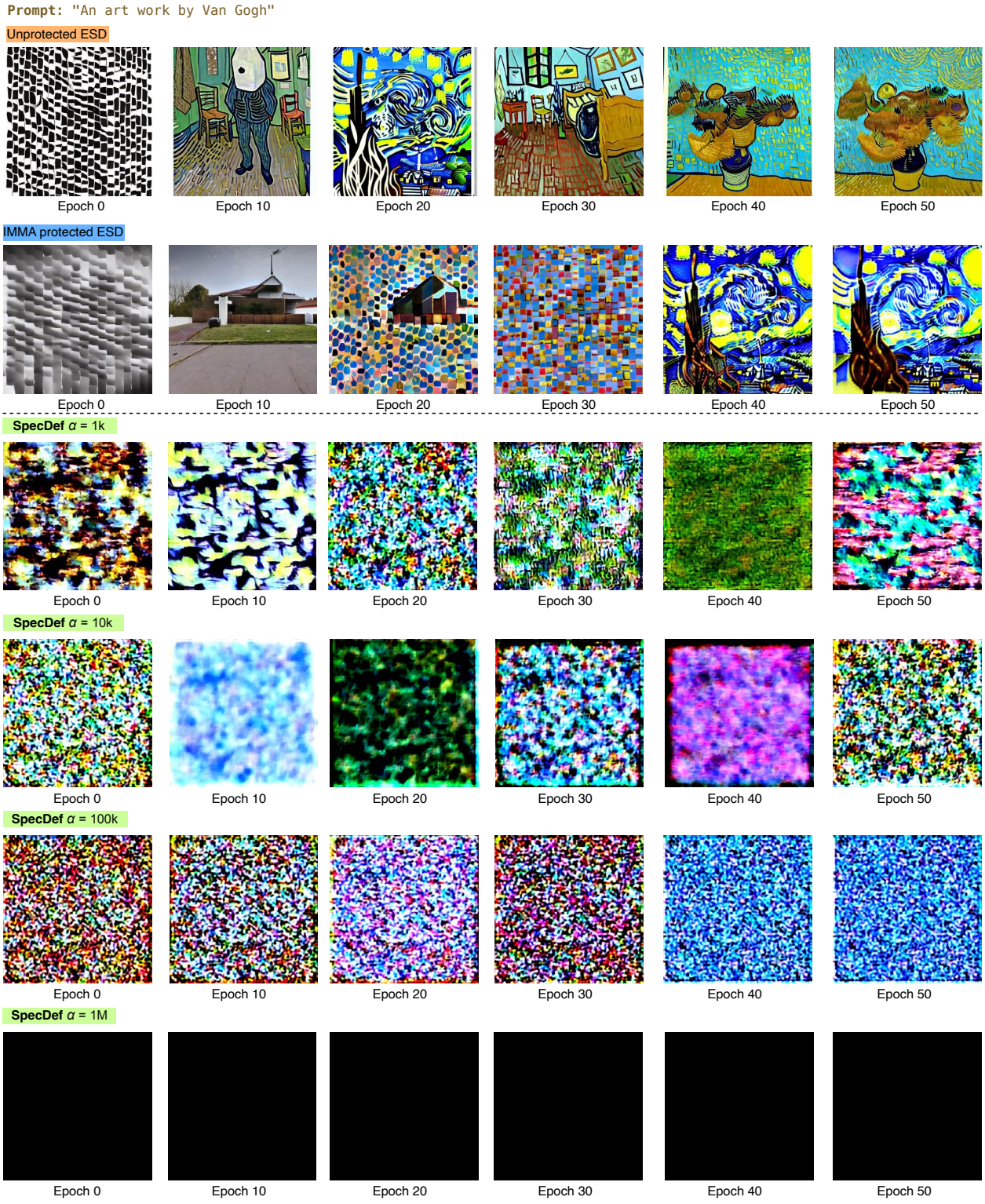


Figure 6. Evolution of generated images over 50 LoRA training epochs for the prompt "An artwork by Van Gogh" under different protection mechanisms, with a LoRA learning rate of 10^{-4}. Both IMMA and SpecDef are applied to all cross-attention layers.

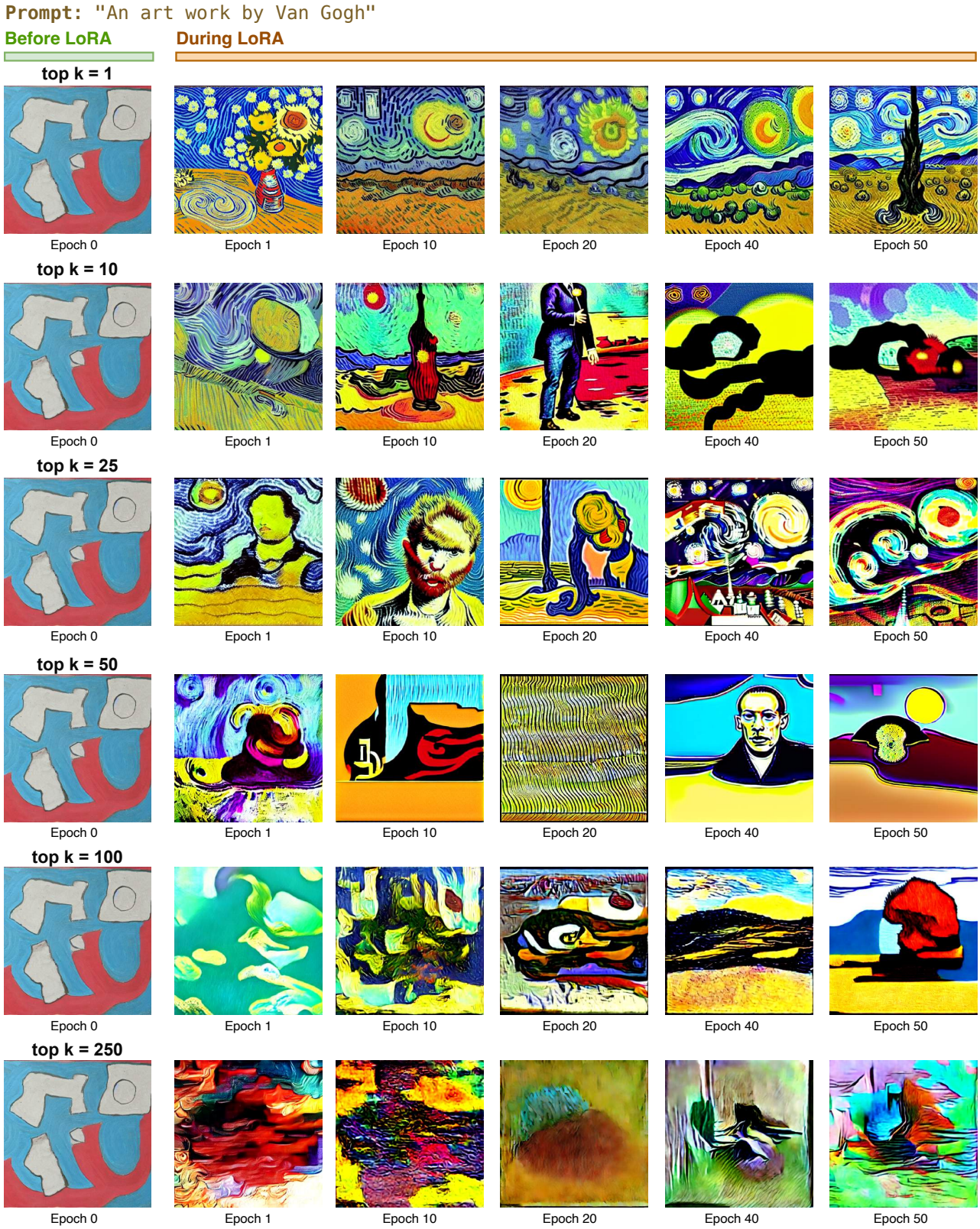


Figure 7. Evolution of generated images over 50 LoRA training epochs for the prompt "An artwork by Van Gogh", with $\alpha = 1000$, a LoRA learning rate of 10^{-4}, and Top-k singular values where $k \in \{1, 10, 25, 50, 100, 250\}$.

Table 19. Van Gogh style re-learning ablations across LoRA learning rates with three random seeds. Rows highlighted in green indicate successful protection against LoRA fine-tuning. Bolded values indicate all outputs collapse to black images across three seeds.

METHOD	SGR_C (CLIP) ↑	SGR_D (DINO) ↑	SGR_L (LPIPS) ↑	RESISTANCE TIME
SPECDEF $\alpha = 100$				
RANDOM 1 LAYER	0.67 ± 0.73%	-4.54 ± 6.92%	-0.99 ± 11.58%	0.46 ± 0.34S
RANDOM 5 LAYERS	1.34 ± 2.53%	-6.45 ± 3.97%	-4.96 ± 2.83%	1.50 ± 0.23S
RANDOM 25 LAYERS	1.62 ± 3.51%	-7.03 ± 4.40%	-5.78 ± 6.65%	10.65 ± 1.15S
ALL 48 CROSS-ATTENTION LAYERS	0.23 ± 2.61%	-9.15 ± 4.80%	-5.90 ± 3.54%	20.27 ± 0.01S
SPECDEF $\alpha = 1,000$				
RANDOM 1 LAYER	0.69 ± 2.46%	-5.50 ± 1.12%	-4.21 ± 5.58%	0.24 ± 0.20S
RANDOM 5 LAYERS	0.42 ± 0.78%	-6.40 ± 6.20%	-4.72 ± 12.29%	1.49 ± 0.23S
RANDOM 25 LAYERS	2.67 ± 1.60%	7.29 ± 12.27%	-0.18 ± 12.20%	10.66 ± 1.14S
ALL 48 CROSS-ATTENTION LAYERS	8.95 ± 0.39%	30.04 ± 4.60%	2.43 ± 10.49%	20.27 ± 0.00S
SPECDEF $\alpha = 10{,}000$				
RANDOM 1 LAYER	0.04 ± 0.52%	-6.44 ± 6.79%	-2.72 ± 12.25%	0.24 ± 0.20S
RANDOM 5 LAYERS	2.78 ± 1.28%	6.36 ± 5.66%	-3.90 ± 11.82%	1.49 ± 0.23S
RANDOM 25 LAYERS	**9.29 ± 0.96%**	**51.40 ± 3.14%**	**42.09 ± 6.33%**	10.66 ± 1.14S
ALL 48 CROSS-ATTENTION LAYERS	**10.81 ± 2.23%**	**44.08 ± 4.08%**	**42.11 ± 21.96%**	20.27 ± 0.00S

Table 20. Van Gogh style re-learning ablations on LoRA learning rate across 3 random seeds. Rows highlighted in green represent successful protection from LoRA fine-tuning. Values in bold font represent all generated images are collapsed to black.

METHOD	SGR_C (CLIP) ↑	SGR_D (DINO) ↑	SGR_L (LPIPS) ↑
REFERENCE IMAGES	0%	0%	0%
IMMA (ZHENG & YEH, 2024)			
10^{-3}	0.65 ± 0.10%	-5.74 ± 5.05%	-0.32 ± 6.21%
10^{-4}	3.34 ± 3.18%	7.47 ± 14.94%	1.77 ± 15.08%
10^{-5}	11.94 ± 1.51%	60.55 ± 3.03%	38.37 ± 8.56%
10^{-6}	10.89 ± 2.90%	52.49 ± 10.75%	35.14 ± 18.60%
10^{-7}	11.61 ± 0.62%	54.64 ± 9.34%	32.45 ± 14.64%
SPECDEF $\alpha = 10$k			
10^{-3}	15.82 ± 2.21%	68.55 ± 11.15%	63.31 ± 4.03%
10^{-4}	10.81 ± 2.23%	44.08 ± 4.08%	42.11 ± 21.96%
10^{-5}	8.31 ± 1.55%	32.91 ± 7.37%	10.91 ± 9.56%
10^{-6}	0.37 ± 1.66%	-7.52 ± 6.10%	-5.37 ± 11.48%
10^{-7}	0.06 ± 1.19%	-8.97 ± 9.37%	-4.71 ± 10.68%
SPECDEF $\alpha = 100$k			
10^{-3}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-4}	10.40 ± 0.54%	44.22 ± 2.00%	49.31 ± 6.14%
10^{-5}	8.18 ± 1.04%	40.43 ± 4.07%	31.79 ± 9.81%
10^{-6}	9.09 ± 1.47%	33.93 ± 5.42%	7.20 ± 7.97%
10^{-7}	0.84 ± 1.44%	-2.01 ± 6.64%	-6.37 ± 11.65%
SPECDEF $\alpha = 1$M			
10^{-3}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-4}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-5}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-6}	11.87 ± 1.22%	50.41 ± 4.90%	64.77 ± 3.72%
10^{-7}	9.26 ± 2.47%	38.56 ± 9.53%	23.05 ± 23.62%
SPECDEF $\alpha = 1B$			
10^{-3}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-4}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-5}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-6}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**
10^{-7}	**17.30 ± 0.31%**	**77.19 ± 1.49%**	**62.83 ± 3.39%**

Table 21. Van Gogh style re-learning ablations on top k singular values across 3 random seeds. Rows highlighted in green represent successful protection from LoRA fine-tuning.

METHOD	SGR_C (CLIP) ↑	SGR_D (DINO) ↑	SGR_L (LPIPS) ↑
REFERENCE IMAGES	0%	0%	0%
SPECDEF $\alpha = 1k$, LORA LEARNING RATE = 10^{-4}			
$k = 1$	0.84 ± 1.07%	-3.39 ± 3.18%	-2.67 ± 9.03%
$k = 5$	-0.36 ± 1.67%	-9.83 ± 2.88%	-9.99 ± 5.84%
$k = 10$	2.99 ± 3.89%	9.11 ± 8.97%	1.37 ± 3.05%
$k = 25$	0.88 ± 1.17%	-5.96 ± 7.00%	-8.05 ± 10.54%
$k = 50$	6.93 ± 1.69%	33.29 ± 11.08%	19.06 ± 5.14%
$k = 100$	4.67 ± 2.38%	15.78 ± 7.43%	7.07 ± 9.66%
$k = 250$	3.70 ± 0.49%	19.47 ± 1.54%	0.53 ± 8.87%

G. Further Ablations and Experimental Variation

The degrees of freedom of our algorithm are very few (number of layers, the multiplier α, and k the number of singular values to choose). Nonetheless, this section provides a number of additional experiments that either support choices made in the paper or supplement other analyses.

G.1. Layers used in Spectral Deformation

In this experiment, we varied the number of layers used for spectral deformation. We repeat the harmful fine-tuning experiment with fresh reruns (different layers may be selected due to seed variation) on `meta-llama_Llama-3.1-8B-Instruct` using $\alpha = 1\text{k}$, which does not provide resistance when layers is 1. Here we show that resistance at a smaller α is possible provided given more layers are used, though for smaller learning rates even more layers or a larger α will need to be used. For spectral deformation with layer injection for all layers it costs up to $2\times$ the original model cost to reparameterized the entire network so choosing a smaller number of layers is more efficient.

Table 22. Harmful fine-tuning results: steps $\pm$ std (harmfulness $\pm$ std). †: some seeds diverged (PPL doubled). ‡: all seeds diverged. Notice that divergence in all seeds only typically occurs when enough layers are used. More layers results in better convergence rate control.

σ	LAYERS	10^{-6}	2×10^{-6}	5×10^{-6}	8×10^{-6}	10^{-5}	3×10^{-5}
1000	1	120 ± 132 (0.53 ± 0.16)†	66 ± 60 (0.54 ± 0.12)†	53 ± 40 (0.56 ± 0.10)†	43 ± 30 (0.64 ± 0.19)†	40 ± 26 (0.62 ± 0.14)†	23 ± 15 (0.63 ± 0.13)†
	2	26 ± 5 (0.62 ± 0.24)†	16 ± 5 (0.59 ± 0.16)†	10 (0.58 ± 0.13)†	10 (0.55 ± 0.16)†	10 (0.59 ± 0.14)†	10 (0.42 ± 0.39)†
	5	20 (0.69 ± 0.10)†	13 ± 5 (0.64 ± 0.15)†	10 (0.60 ± 0.06)‡	10 (0.59 ± 0.12)‡	10 (0.47 ± 0.39)‡	10 (0.25 ± 0.27)‡
	10	16 ± 5 (0.72 ± 0.05)†	13 ± 5 (0.68 ± 0.18)‡	10 (0.67 ± 0.05)‡	10 (0.44 ± 0.38)‡	10 (0.43 ± 0.37)‡	10 (0.03 ± 0.05)‡
5000	1	83 ± 110 (0.55 ± 0.10)†	60 ± 62 (0.58 ± 0.12)†	40 ± 36 (0.60 ± 0.12)†	36 ± 30 (0.64 ± 0.17)†	30 ± 26 (0.64 ± 0.15)†	23 ± 15 (0.59 ± 0.21)†
	2	10 (0.58 ± 0.13)†	10 (0.56 ± 0.16)†	10 (0.48 ± 0.28)†	10 (0.41 ± 0.40)†	10 (0.42 ± 0.41)†	10 (0.41 ± 0.41)†
	5	10 (0.59 ± 0.08)‡	10 (0.44 ± 0.33)‡	10 (0.32 ± 0.24)‡	10 (0.45 ± 0.34)‡	10 (0.10 ± 0.08)‡	10 (0.01 ± 0.02)‡
	10	10 (0.65 ± 0.07)‡	10 (0.44 ± 0.38)‡	10 (0.12 ± 0.17)‡	10 (0.00 ± 0.00)‡	10 (0.00 ± 0.00)‡	10 (0.04 ± 0.05)‡
10000	1	83 ± 110 (0.57 ± 0.11)†	60 ± 70 (0.59 ± 0.13)†	40 ± 36 (0.60 ± 0.13)†	30 ± 26 (0.56 ± 0.19)†	30 ± 26 (0.56 ± 0.15)†	16 ± 11 (0.56 ± 0.13)†
	2	10 (0.55 ± 0.16)†	10 (0.58 ± 0.18)†	10 (0.43 ± 0.39)†	10 (0.40 ± 0.42)†	10 (0.37 ± 0.41)†	10 (0.35 ± 0.36)‡
	5	10 (0.64 ± 0.07)‡	10 (0.38 ± 0.35)‡	10 (0.25 ± 0.22)‡	10 (0.06 ± 0.05)‡	10 (0.00 ± 0.00)‡	10 (0.03 ± 0.04)‡
	10	10 (0.40 ± 0.36)‡	10 (0.22 ± 0.29)‡	10 (0.01 ± 0.02)‡	10 (0.12 ± 0.15)‡	10 (0.01 ± 0.02)‡	10 (0.09 ± 0.13)‡

We additionally ran an ablation study where we select different layers for spectral deformation to show which layers are effective and which are not. We select a single layer ℓ from $\ell \in \{2, 16, 30\}$ that represent early, middle and later parts of the model. With the same experimental settings above and $\alpha = 10\text{k}$. The experiments were run over 3 random seeds. We selected one of the following seven components for each experiment: (1) a Query matrix, (2) a Key matrix, (3) a Value matrix, (4) an Output projection, (5) a Gated projection, (6) an Up projection, or (7) a Down projection. When looking at effectiveness by position, *averaged over all seven components*, there was not a large difference between layer depths: early layers ($\ell = 2$) achieved a mean harmfulness score of 0.40 ± 0.37, middle layers ($\ell = 16$) scored 0.32 ± 0.29, and late layers ($\ell = 30$) scored 0.30 ± 0.32. These means are close relative to their standard deviations, indicating that layer position alone is not the dominant factor when averaging across component types.

However, for specific components, effectiveness can vary sharply with depth. The best components to select were the output projections after attention, which achieved very low harmfulness across all depths (e.g., $\ell = 30$: 0.0014, $\ell = 16$: 0.0598, $\ell = 2$: 0.0021). The value matrices were similarly effective: $\ell = 30$ harmfulness is 0.0042, $\ell = 16$ harmfulness is 0.0038, and $\ell = 2$ harmfulness is 0.0122. In contrast, other components showed strong depth dependence. For example, the MLP up projection had low harmfulness at early layers ($\ell = 2$: 0.0864) but was largely ineffective at later depths ($\ell = 16$: 0.5593, $\ell = 30$: 0.5488). Gate and down projections were generally ineffective in early layers but showed some effectiveness in later layers.

Query and key matrices (as well as up and down projections) are often very rank deficient, which can be predicted from the results in Table 10 where Theorem 3 becomes loose. The modest trend toward later layers being more effective than earlier layers (in the component-averaged means), and why some types of layers are more effective than others may also be explained by how often the layer appears as a factor in the Hessian blocks.

G.2. Top K Singular Values for Deformation

In order to use SpecDef, we need to select the top-k singular values. In this section, we will provide an empirical analysis on the impact of that choice in Section G.2. We purposely choose a lower α than is effective to illustrate the effects of scaling k and we chose to use the original base model `Llama-3.1-8B-Instruct` which is already close performing well at this

task. This was done to make training resistance as hard as possible for SpecDef to highlight the effect of varying k.

k	lr=10^{-6}	lr=5×10^{-6}	lr=10^{-5}	lr=3×10^{-5}
1	330±180 (0.573)	50±0 (0.616)	30±0 (0.614)	20±10 (0.514)†
2	350±160 (0.578)	55±5 (0.602)†	30±0 (0.590)†	10±0 (0.433)‡
5	400±110 (0.594)	45±5 (0.594)†	30±0 (0.592)†	25±15 (0.443)‡
10	325±185 (0.571)	45±5 (0.594)†	35±5 (0.576)†	15±5 (0.475)‡
25	340±170 (0.588)	50±20 (0.541)†	30±10 (0.531)‡	15±5 (0.455)‡
all	325±185 (0.573)	30±0 (0.451)‡	20±0 (0.439)‡	10±0 (0.398)‡

Table 23. Top-k ablation ($\alpha = 10,000$). Number of steps and WMDP classification accuracy (mean±std over 3 seeds). †: some seeds doubled perplexity; ‡: all seeds doubled perplexity.

Our results in Section G.2 are that as SpecDef uses more k divergence occurs faster and SpecDef is able to provide resistance for smaller learning rates.

H. Are Previous Resistance Methods Convergence-Rate Control in L?

In this section, we examine how existing resistance methods relate to our framework. Prior work is motivated by heterogeneous formalisms—ranging from information-theoretic arguments (Rosati et al., 2024b) to metalearning perspectives (Tamirisa et al., 2025). A comprehensive unification connection each directly to our framework is out of scope of this paper. Instead, we ask a sharply defined question aligned with our theory:

Do existing resistance methods implicitly control convergence by manipulating curvature, e.g., via the Lipschitz constant L?

To this end, we conduct a layerwise spectral analysis of defended models.

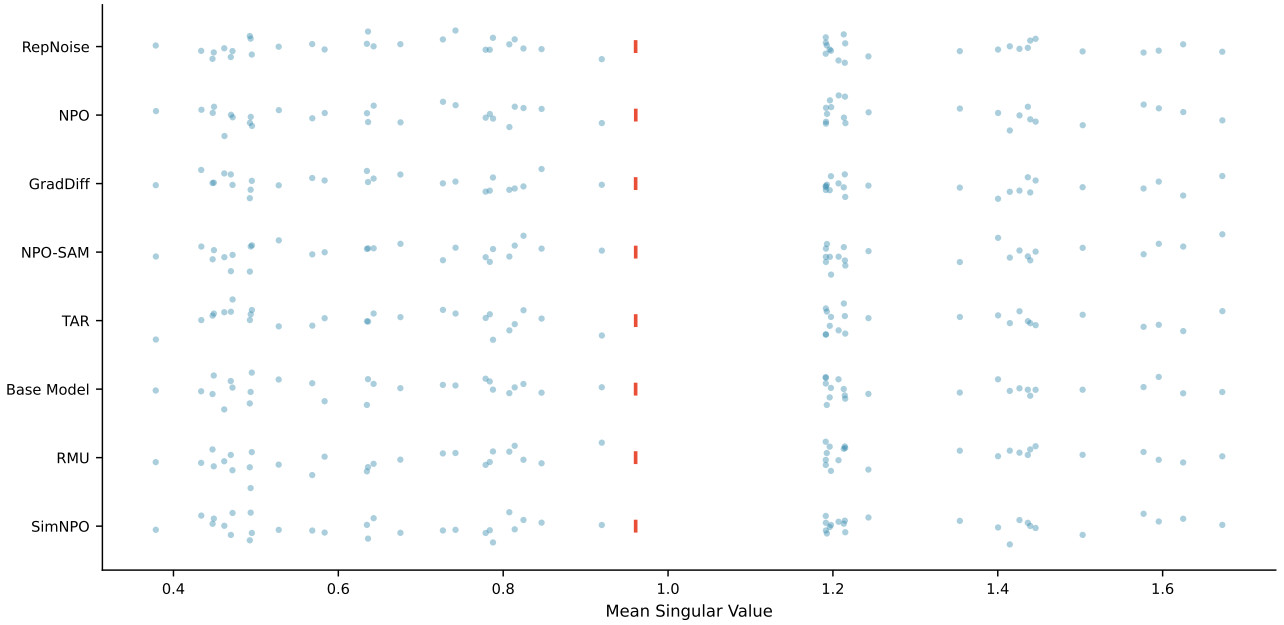


Figure 8. Mean singular value of each layer's weight matrix across resistance methods (rows). Red bars indicate layerwise means. SpecDef is omitted since its reparameterized layers have singular values that are orders of magnitude larger by construction. Or finding is that weight spectra is not meaningfully different from the base model.

Observation: No Curvature Inflation As shown in Figure 8, all methods exhibit weight spectra closely matching the base model (`Meta-Llama-3-8B-Instruct`). No method exceeds $\sigma_1(\theta_i) \approx 15$, and the variance of singular values is small (typically $\sim$0.45), indicating uniformly good conditioning. Thus, these resistance methods do *not* implement convergence-rate control by increasing L through enlarged spectral norms, nor other kinds of numerical instability through

weight-based poor conditioning. This is not surprising given that each method is built using a loss function and trained using standard proceedures such as the use of Adam and the use of auxiliary retain loses which would prevent poor conditioning and inflated spectral values.

This aligns with their empirical behaviour: all methods remain trainable under sufficiently strong learning rates and do not exhibit the divergence characteristic of large-L systems (see Table 1). Moreover, several methods can still be fine-tuned on unrelated objectives (Rosati et al., 2024a), which would be inconsistent with global curvature inflation and is orthogonal to the mechanism we propose.

Locally Flat Curvature via Alignment Rather Than Magnitude. One might instead hypothesize that these resistance methods slow optimization by introducing unusually small curvature directions, increasing the number of gradient steps required for curvature-unaware optimizers. However, we do not observe systematically smaller singular values than in the base model. In fact, gradient norms are often *initially larger*, as shown in Figure 9.

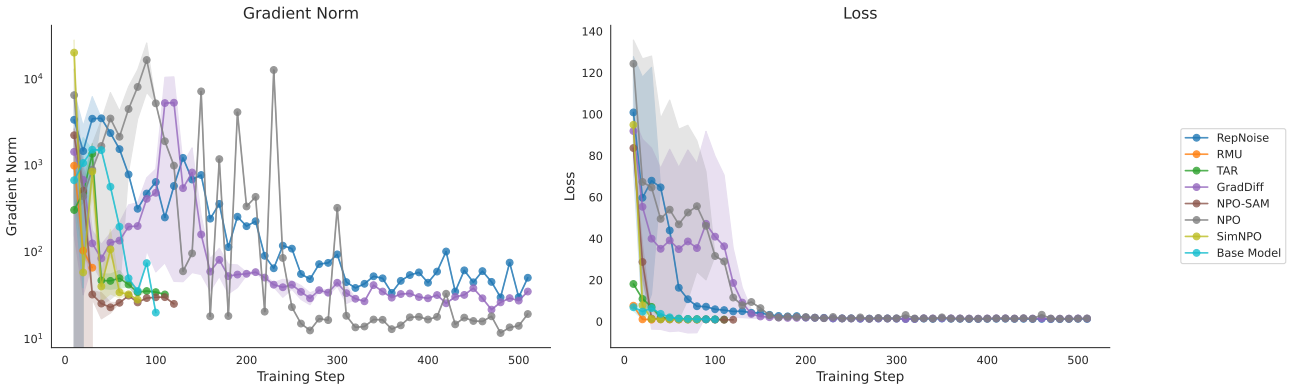


Figure 9. Training dynamics across resistance methods, showing loss and log gradient norm. On successful resistance runs `RepNoise`, `NPO`, and `GradDiff` enter a small gradient norm regime where loss does not change.

This combination—unaltered spectral magnitudes but large initial gradient norms—points to a more subtle mechanism: *spectral alignment*. Within our framework, flat curvature can arise when activation Jacobian align with singular vectors corresponding to small singular values; this produces loss-specific flatness without globally modifying L. Flat curvature also results in convergence rate control when a chosen learning rate $1/L$ is not large enough to produce effective gradient steps in the flat direction. In cases where the learning rate cannot be increased since there are other large curvature directions simply increasing the learning rate would cause divergence.

Crucially, such flatness is fragile. Unlike our k-root layer injection, flatness can be undone by explicitly sharpening curvature. In particular, increasing the largest singular values via spectral reparameterization immediately destroys this alignment-induced flatness. We should be clear that the curvature increase still needs to be in the sub-divergence regime with only a small α.

Spectral Reparameterization as an Attack To test this hypothesis, we apply SpecDef to 20 layers of each defended model with $\alpha = 10$ ($\alpha = 1$ baseline). The results in Table 24 are striking: resistance methods that were previously effective consistently fail once curvature is increased. We refer to this as a *spectral reparameterization attack*.

These results suggest that many existing resistance methods rely on loss-specific spectral alignment rather than L-based curvature control. From this perspective, our framework does not rule out these resistance methods as a class; rather, it explains both their apparent effectiveness and their brittleness. It also suggests a principled path forward: explicitly constructing resistance methods that control singular *vector* alignment, not merely spectral magnitudes. Since we saw that pervious methods often pair well with SpecDef in the high α regime, future training resistance methods may be combined with SpecDef to provide an even higher cost to undo defences.

Curvature-aware Optimization as an Attack Table 25 shows that Sophia or AdaHessian is very effective (often more so than Adam) in undoing previous defence resistance. Muon was not very effective, which further corroborates that these methods may not operate by poor conditioning which Muon is effective at undoing. Instead, the results from Sophia and

Table 24. **Spectral Reparameterization Attacks.** Comparison of defended models under $\alpha = 1$ vs. $\alpha = 10$. Entries report mean training steps $\pm$ std, with final accuracy $\pm$ std in parentheses. † indicates at least one divergent run (perplexity doubled); ‡ indicates all runs diverged.

RESISTANCE METHOD	$\alpha = 1$	$\alpha = 10$
REPNOISE	510 (0.459)	26 ± 5 (0.630 ± 0.022)
RMU	510 (0.573)	23 ± 5 (0.618 ± 0.013)
TAR	510 (0.302)	70 ± 45 (0.510 ± 0.170)†
GRADDIFF	510 (0.409 ± 0.002)	196 ± 120 (0.609 ± 0.010)
NPO-SAM	510 (0.412 ± 0.000)	70 ± 43 (0.610 ± 0.008)
NPO	510 (0.553)	73 ± 15 (0.616 ± 0.016)
SIMNPO	510 (0.518)	56 ± 25 (0.625 ± 0.030)
LLAMA-3-8B	100 (0.608)	20 (0.631 ± 0.012)

AdaHessian seem to point to flat curvature induced by the training objective as how they operate since second-order methods are designed to account for both high negative and positive curvature.

Table 25. Curvature-aware optimizer attacks on defended models ($\alpha = 1$). Entries report training steps to reach target metric (mean $\pm$ std across seeds). †: some seeds diverged; ‡: all seeds diverged. Unlike with SpecDef, these attacks are often much more successful on defences that were previously able to resist Adam.

	MUON			SOPHIA			ADAHESSIAN		
MODEL	10^{-5}	2×10^{-5}	5×10^{-5}	5×10^{-6}	10^{-5}	2×10^{-5}	10^{-6}	3×10^{-6}	5×10^{-6}
BASE MODEL	$765\ (0.52 \pm 0.04)$	$653 \pm 193\ (0.54 \pm 0.05)$	$190 \pm 62\ (0.62 \pm 0.01)$	$26 \pm 5\ (0.61 \pm 0.01)$	$20\ (0.65 \pm 0.05)$	$16 \pm 5\ (0.63 \pm 0.04)$	$146 \pm 25\ (0.63 \pm 0.01)$	$60\ (0.61 \pm 0.01)$	$56 \pm 5\ (0.62 \pm 0.01)$
NPO	765 (0.29)	765 (0.30)	765 (0.55)	60 (0.69)	60 (0.54)‡	40 (0.42)‡	$467 \pm 420\ (0.60 \pm 0.02)$	$140 \pm 70\ (0.61 \pm 0.01)$	$120 \pm 42\ (0.60 \pm 0.00)$
GRADDIFF	$765\ (0.25 \pm 0.03)$	$765\ (0.24 \pm 0.03)$	$765\ (0.44 \pm 0.04)$	$73 \pm 15\ (0.63 \pm 0.03)$	$93 \pm 5\ (0.62 \pm 0.02)$	$56 \pm 15\ (0.42 \pm 0.01)^{\ddagger}$	$553 \pm 366\ (0.56 \pm 0.04)$	$313 \pm 210\ (0.60 \pm 0.00)$	$183 \pm 90\ (0.60 \pm 0.01)^{\dagger}$
RMU	$765\ (0.23 \pm 0.02)$	$765\ (0.30 \pm 0.03)$	$633 \pm 228\ (0.58 \pm 0.02)$	$46 \pm 11\ (0.65 \pm 0.04)$	$50 \pm 43\ (0.63 \pm 0.01)$	$33 \pm 15\ (0.54 \pm 0.12)^{\dagger}$	$183 \pm 60\ (0.61 \pm 0.01)$	$76 \pm 11\ (0.63 \pm 0.03)$	$70 \pm 10\ (0.62 \pm 0.02)$
TAR	$765\ (0.30 \pm 0.02)$	$765\ (0.30 \pm 0.03)$	$435 \pm 285\ (0.29 \pm 0.02)^{\dagger}$	$86 \pm 5\ (0.62 \pm 0.02)$	$93 \pm 15\ (0.58 \pm 0.06)^{\dagger}$	$33 \pm 11\ (0.34 \pm 0.04)^{\ddagger}$	$110 \pm 26\ (0.25 \pm 0.06)^{\ddagger}$	$46 \pm 11\ (0.28 \pm 0.05)^{\ddagger}$	$33 \pm 5\ (0.25 \pm 0.00)^{\ddagger}$
RR	$765\ (0.23 \pm 0.02)$	$765\ (0.41 \pm 0.03)$	$383 \pm 119\ (0.60 \pm 0.00)$	$36 \pm 5\ (0.61 \pm 0.01)$	$36 \pm 11\ (0.65 \pm 0.06)$	$43 \pm 11\ (0.49 \pm 0.13)^{\dagger}$	$140 \pm 26\ (0.61 \pm 0.01)$	$70\ (0.62 \pm 0.02)$	$56 \pm 5\ (0.62 \pm 0.01)$
REPNOISE	$765\ (0.31 \pm 0.03)$	$765\ (0.23 \pm 0.04)$	410 (0.60)	40 (0.66)	30 (0.67)	30 (0.62)	$65 \pm 7\ (0.63 \pm 0.03)$	$55 \pm 21\ (0.63 \pm 0.01)$	$45 \pm 7\ (0.63)$

Alternative Explanation: Stochastic Obfuscation A second possibility, following Athalye et al. (2018), is that these resistance methods increase the variance term governing stochastic convergence by biasing minibatch gradients, thereby violating the unbiased-gradient assumption underlying SGD. We tested this hypothesis via gradient accumulation (factor 10) and increased batch size (64). Neither intervention materially weakened the resistance in most methods, whereas curvature sharpening did. While this does not fully rule out stochastic effects, it suggests they are not the primary mechanism. For `DeepIgnorance` with strong filtering (O'Brien et al., 2025), we did find that these attacks did meaningfully change the resistance strength: at a 10^{-5} learning under our usual first-order attack the model could not be trained for WMDP-Bio. By applying a gradient accumulation of 10 and batch size of 16, the model reached over 60% accuracy in around 200 training steps. This indicates that pre-training based methods might have obfuscated gradients w.r.t./ a harmful training objective. `DeepIgnorance` with a strong filter was similarly easily undone using curvature-aware methods but we did not report these since our goal is to compare defences applied to the same model and `DeepIgnorance` is itself a newly pretrained model.

Takeaway In summary, existing resistance methods do not significantly alter weight spectra. Their behaviour is most consistent with loss- and data-local spectral alignment that produces locally flat curvature directions. This observation both motivates our approach and highlights a future research direction: understanding how singular vector rotation, Jacobian spectra, and data alignment jointly shape effective curvature so that more robust resistance methods can be built.

I. Layer Injection Attack Mathematical Details

In this appendix, we will provide proofs and various remarks for the k-th layer injection attack as well as the universality result that all convergence rate control methods are spectral reparamaterization subject to this attack.

I.1. Proof of k-th layer injection attack (Theorem 8)

First, we provide a proof for Theorem 8 in Section I.1. Restating the Theorem:

Theorem 32 (k-th layer injection attack)**.** *Consider a feed-forward network*

$$f(x) = \theta_{n+1} \circ \phi_n \circ \theta_n \circ \cdots \circ \phi_1 \circ \theta_1 x,$$

where each $\theta_i \in \mathbb{R}^{d_i \times d_{i-1}}$ is a linear layer and each ϕ_i is a fixed, elementwise activation. Let $M = \max_{1 \le i \le n} \sigma_1(\theta_i)$. Then to achieve $\max_i \sigma_1(\theta_i) \le M^{1/k}$ it suffices to insert at most $(k-1) \cdot n$ additional linear layers and perform spectral deformation.

Proof. Assume the defender would engage in the best possible resistance:

$$\sigma_1(\theta_1) = \cdots = \sigma_1(\theta_n) = c,$$

where c is large enough to resist convergence. From the theorem statement c = M. Since otherwise, the attack is less expensive (less layers to inject).

By Theorem 6, it suffices for only one weight to have $\sigma_1(\theta_i) = c$ to achieve spectral control over H_θ^f. This implies that, assuming a defender has made the network non-convergent, to achieve convergence the attacker must ensure that

$$\max\{\sigma_1(\theta_1), \ldots, \sigma_1(\theta_n)\} \le r.$$

where r is selected to so that $1/r$ is a reasonable learning rate that gives convergence.

To reduce $\sigma_1(\theta_i)$, we can use a procedure analogous to our spectral deformation algorithm. Let

$$\tilde{\Sigma}_{\theta_i} \leftarrow T\Sigma_{\theta_i}, \quad T = \operatorname{diag}\big(r/\sigma_1(\theta_i), 1, \ldots, 1\big),$$

and reconstruct

$$\tilde{\theta}_i = U_{\theta_i} \tilde{\Sigma}_{\theta_i} V_{\theta_i}^\top.$$

Without loss of generality, we assume that $\sigma_2(\theta_i) < r$ (otherwise T can be modified accordingly). For simplicity, we assume the network uses activation functions that are non-homogeneous, so compensation can only occur within layers, not across layers. Therefore, we inject the following compensation matrix after $\tilde{\theta}_i$:

$$\tilde{I}_i = U_{\theta_i} \Sigma_{\theta_i} \tilde{\Sigma}_{\theta_i}^{-1} U_{\theta_i}^\top,$$

that is, we now have

$$f(x) = \theta_{n+1} \circ \phi_n \circ \theta_n \circ \cdots \circ \phi_i \circ \tilde{I}_i \tilde{\theta}_i \circ \cdots \circ \phi_1 \circ \theta_1 x$$

Observe that by compensation

$$\sigma_1(\tilde{\theta}_i) = r, \qquad \sigma_1(\tilde{I}_i) = \sigma_1(\theta_i)/r.$$

We now seek the r^* that minimizes the worst-case singular value. We have the following equality by definition:

$$r^* = \arg\min_r \max\{\sigma_1(\tilde{\theta}_i), \sigma_1(\tilde{I}_i)\} = \arg\min_r \max\{r, \sigma_1(\theta_i)/r\}.$$

Since the first term of the max is increasing in r and the second term is decreasing, it is well-known that the minimum occurs at their intersection, when

$$r = \frac{\sigma_1(\theta_i)}{r},$$

which works out to

$$r^* = \sqrt{\sigma_1(\theta_i)}.$$

This is the maximum achievable reduction when injecting a single compensating layer, yielding

$$\max\{\sigma_1(\tilde{\theta}_i), \sigma_1(\tilde{I}_i)\} = \sqrt{\sigma_1(\theta_i)}.$$

For m injected compensating layers, we have $m+1$ total matrices in the factorization: the deformed layer $\tilde{\theta}_i$ and m compensation matrices $\tilde{I}_i^{(1)}, \ldots, \tilde{I}_i^{(m)}$. These must satisfy

$$\tilde{I}_i^{(m)} \cdots \tilde{I}_i^{(1)} \tilde{\theta}_i = \theta_i,$$

so the product of their singular values is constrained by $\sigma_1(\theta_i)$. To minimize the maximum singular value across all $m+1$ matrices, we set them equal. If each has spectral norm r, the constraint becomes $r^{m+1} \geq \sigma_1(\theta_i)$ and hence $r \geq \sigma_1(\theta_i)^{1/(m+1)}$ for the best possible worst-case singular value, giving optimal factorization

$$r^* = \sigma_1(\theta_i)^{1/(m+1)},$$

which yields $\sqrt[k]{\sigma_1(\theta_i)} = \sqrt[k]{c}$ for the given layer θ_i where $k = m+1$. This is achievable by using SVD and distributing the singular values across the injected layers.

So in order to achieve $\max\{\sigma_1(\theta_i)\}_i^{n\cdot(k-1)} = \sqrt[k]{c}$, we need $(k-1)\cdot n$ layers since we need to reduce the singular values for *each* of the n original layers. This concludes the proof. □

I.2. Layer injection is a Strong Universal Attack

In this section we show that the layer injection attack presented in § 6 is an upper-max spectral reparameterization (Theorem 34), which we define below (Theorem 33). We also show why this attack would be universal under spectral reparameterizations of deep networks (Theorem 36). The implication of these two results is that this attack is a strong adaptive attack in the sense that if the attacker knows the curvature was manipulated to control convergence, then under relatively mild assumptions specified in Theorem 36, the curvature-based convergence control will be undone (though possibly at a cost that is too expensive for the attacker).

First, let us provide a general definition of l,u,k-spectral reparameterization.

Definition 33. l,u,k-Spectral Reparameterization Given function f parameterized by θ and a constant $l, u > 0$, $\epsilon \geq 0$, and $0 < k \leq n$. A map $\mathcal{T}_{u,l} : f_\theta \to f_{\theta'}$ is a l,u,k spectral reparameterization.

1. **Spectral Control** $l \leq \sigma_k(H_\theta) \leq u$ where H_θ is the Hessian w.r.t. the parameters of f_θ.
2. **Functional Invariance up to** ϵ There exists distance function $d(f, g)$ on the space of functions $f, g \in \mathcal{F}$ such that $d(\mathcal{T}_{u,l}[f], f) \leq \epsilon$ for a small $\epsilon \geq 0$.

An upper-max spectral reparameterization is a 0,c,1-Spectral Reparameterization where c is a constant that provides an upper bound for the maximum singular values.

Theorem 34 (Layer Injection is Upper-Max Spectral Reparameterization)**.** *Layer injection attacks with sufficiently selected layers are an upper-max spectral reparameterization.*

Proof. To prove this we need to show two things (1) functional invariance and (2) spectral control.

(1) Functional invariance Functional invariance proof is the exact same as that of the proof for Theorem 6.

(2) Spectral Control

First, we show that

$$\sigma_1(H_\theta) \leq \sum_{i,j} \sigma_1\left(\frac{\partial^2 f}{\partial\theta_i \partial\theta_j}\right).$$

This follows from the application of the triangle inequality of the spectral norm $\|\cdot\|_2$ to the following block decomposition.

Any given matrix M admits the following sum over its disjoint submatrices

$$M = \begin{bmatrix} A_{1,1} & \dots & A_{1,m} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \dots & A_{n,m} \end{bmatrix} = \begin{bmatrix} A_{1,1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} + \cdots + \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & A_{n,m} \end{bmatrix}.$$

Suppose we had the Hessian matrix $H_\theta \in \mathbb{R}^{n\times n}$. Let $H^0_{i,j} \in \mathbb{R}^{n\times n}$ be the matrix $\frac{\partial^2 f}{\partial\theta_i\partial\theta_j}$ embedded in a 0 matrix in the original position i, j it would have been in H. Here i, j index the blocks of H_θ corresponding to the vectorized parameter matrices θ_i and θ_j where each block may itself be a matrix.

Now it follows that

$$H_\theta = \sum_{i,j}^{n} H^0_{i,j}.$$

Using the subadditivity of operator norms, we see that the spectral norm and hence σ_1 has the following property:

$$\sigma_1(H) = \|H\|_2 = \left\| \sum_{i,j} H^0_{i,j} \right\|_2 \leq \sum_{i,j} \left\| H^0_{i,j} \right\|_2$$

Since embedding a matrix in a larger 0 matrix does not change the spectral norm, we have that $\left\| H^0_{i,j} \right\|_2 = \left\| \frac{\partial^2 f}{\partial \theta_i \partial \theta_j} \right\|_2 = \sigma_1 \left(\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} \right)$. Putting this together proves that

$$\sigma_1(H_\theta) \leq \sum_{i,j} \sigma_1 \left(\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} \right).$$

Next all we need to show is that assuming at least one Hessian block $\frac{\partial^2 f}{\partial \theta_i \partial \theta_j}$ admits the algebraic structure $A\theta_k B$, we can control the upper bound on $\sigma_1(H_\theta)$ established above by manipulating $\sigma_1(\theta_k)$. Like in the other proofs, we assume that this structure exists, which has been shown in many examples throughout the paper for neural networks. Now we just directly apply the upper bound of Theorem 17. Yielding

$$\begin{aligned} \sigma_1 \left(\frac{\partial^2 f}{\partial \theta_i \partial \theta_j} \right) &= \sigma_1 (A\theta_k B) && \text{(By assumption)} \\ &\leq \sigma_1(A)\sigma_1(\theta_k)\sigma_1(B). && \text{(By Theorem 17)} \end{aligned}$$

Taken together, this shows that by reducing $\sigma_1(\theta_i)$ we have some upper spectral control of $\sigma_1(H^f_\theta)$. To obtain the upper bound u, we must select a sufficient number of layers $i = k$ such that the upper bound sum we showed earlier is not dominated by some layer θ_p that is not used. We now be sure that u can be chosen based on the desired maximum singular value after layer injection, as demonstrated in Theorem 8. □

We now show an impossibility result. Under weak assumptions that most deep learning settings satisfy, convergence rate control must be a spectral reparameterization using parameter matrices alone (Theorem 36). Combined with Theorem 34, which establishes that the layer injection attack is an upper-max spectral reparameterization, this implies that any lower-max spectral resistance method operating on weights can be countered: the defender must increase $\sigma_1(H^f_\theta)$ through weight manipulation, and the attacker can subsequently decrease it while preserving functionality. Thus, no spectral reparameterization of parameter matrices exists that cannot be undone by our attack.

We first state a lemmas that will help us in our proof.

Lemma 35. *The spectral norm of an outer product of vectors $u, v \in \mathbb{R}^d$ is given by*

$$\|uv^\top\|_2 = \|u\|_2 \|v\|_2.$$

Proof. Consider the following variational characterization of the spectral norm

$$\|A\|_2 = \max_{\|x\|_2 = 1} \|Ax\|_2.$$

For an outer product $uv^\top$, we have

$$\begin{aligned} \|uv^\top\|_2 &= \max_{\|x\|_2=1} \|u(v^\top x)\|_2 \\ &= \max_{\|x\|_2=1} |v^\top x| \|u\|_2 && \text{(absolute homogeneity)} \\ &= \|u\|_2 \|v\|_2, && \text{(by Cauchy-Schwarz, max at } x = v/\|v\|_2) \end{aligned}$$

as required. □

Theorem 36 (Only Weight Matrices Provide Unbounded Spectral Control)**.** *Suppose we have the following feed forward network*

$$f(x) = \theta_{n+1} \circ \phi_n \circ \theta_n \circ \cdots \circ \phi_1 \circ \theta_1 x,$$

with element-wise activation functions $\phi_i : \mathbb{R}^d \to \mathbb{R}^d$ and parameter matrices $\theta_i \in \mathbb{R}^{d_i \times d_{i-1}}$. Further, suppose that the activation functions ϕ were approximately non-expansive and had non-expansive derivatives and second derivatives if they exist. Also, let the data be bounded with $\|x\|_2 \leq c$ for any x drawn from the dataset $\mathcal{D}$.

A transformation of that function $\mathcal{T} : \mathcal{F} \to \mathcal{F}$ is a lower max spectral reparameterization (i.e., a $(l, \infty, 1)$-spectral reparameterization) only if the transformation is a reparameterization of the parameter matrices θ_i.

Proof. The reverse implication (only if) will be proven by showing that the function composition we assumed, our deep neural architecture, has only one unbounded component, the parameter matrices θ_i which allows us to achieve an arbitrary l with which $\sigma_1(H^f_\theta) \geq l$.

Recall that one condition of lower max spectral reparameterization was that the transformation must be able to achieve

$$\sigma_1(H^f_\theta) \geq l,$$

for any l.

Suppose that the transformation involved sampling an x to control convergence in such a way while respecting, by our earlier assumption, that $\|x\| \leq c$. Recall that we had the block decomposition such that $\sigma_1(H^f_\theta) \geq \sigma_1(\frac{\partial^2 f}{\partial \theta_i \partial \theta_j})$. Since x appears in the function composition, these Hessian blocks will consist of outer products involving x or Kronecker products involving x to maintain the matrix structure of the block.

For outer products, we have blocks of the form $Aux^\top B$ for weight matrices A, B and some intermediate vector u. By Theorem 35,

$$\sigma_1(Aux^\top B) \leq \sigma_1(A)\|u\|\|x\|\sigma_1(B) \leq c \cdot \sigma_1(A)\|u\|\sigma_1(B).$$

For Kronecker products, we have blocks of the form $A(x^\top \otimes U)B$. Since spectral norms are multiplicative under Kronecker products, $\sigma_1(x^\top \otimes U) = \|x\|\sigma_1(U)$, and thus

$$\sigma_1(A(x^\top \otimes U)B) \leq \sigma_1(A)\|x\|\sigma_1(U)\sigma_1(B) \leq c \cdot \sigma_1(A)\sigma_1(U)\sigma_1(B).$$

In both cases, the contribution of x to σ_1 of any Hessian block is bounded above by c times a constant K depending only on the weight matrices. Therefore, $\sigma_1(H^f_\theta) \leq c \cdot K$ for some $K > 0$ independent of the data. For any target $l > c \cdot K$, manipulation of the data alone cannot achieve $\sigma_1(H^f_\theta) \geq l$, contradicting the requirement of a lower-max spectral reparameterization.

Suppose that the transformation involved selecting an activation function under the assumptions above. We have seen that by Theorem 3 for $\sigma_1(\frac{\partial^2 f}{\partial \theta_i \partial \theta_j})$, we have

$$\sigma_{r_1}(A)\sigma_1(B)\sigma_{r_2}(C)\cos\theta(A_{r_1}, BC_1)\cos\theta(B, C_{r_2}) \leq \sigma_1(\nabla^2_\theta \mathcal{L}).$$

In this case, B involves some products containing either D'_{z_i} or D''_{z_i} rather than θ_k. Since the activation functions are assumed to be element-wise approximately non-expansive and to have approximately non-expansive first and second derivatives, we have

$$|\phi'_i(t)| \lesssim 1 \text{ and } |\phi''_i| \lesssim 1 \quad \text{for all } t \in \mathbb{R}.$$

Consequently, the diagonal matrices D'_{z_i} and D''_{z_i}, which embed these derivatives at the pre-activations z_i, have the operator norm uniformly bounded by one:

$$\|D'_{z_i}\|_2 \lesssim 1, \quad \|D''_{z_i}\|_2 \lesssim 1.$$

Therefore, any Hessian block involving activation derivatives contributes at most a constant (independent of the weights) to $\sigma_1(H^f_\theta)$. In particular, manipulating the activation functions alone cannot produce arbitrarily large singular values of the Hessian, and thus cannot satisfy the lower max spectral reparameterization condition for sufficiently large ℓ. For piece-wise linear activations that may not be twice differentiable case it is even more true they cannot contribute unbounded increases

to the Hessian singular values. Due to this contradiction manipulation of the activation functions alone cannot be a spectral reparameterization.

The only remaining possibility is manipulation of weights θ_i which we have already shown is a spectral reparameterization in Theorem 6. □

Remark 37 (Applicability to Modern Deep Networks). The bounded data assumption $\|x\|_2 \leq c$ is mild in practice. For images, pixel normalization ensures boundedness. For language models, token embeddings are bounded by construction. Even when data distributions have infinite support, training datasets have finite maximum norms. Non-elementwise functions such as softmax and normalization are also bounded or have a small Lipschitz factor such that using them to control convergence rate is infeasible. Counter-examples may exist outside of our boundedness assumptions but they would take the form of a very exotic architecture. These assumptions are common (e.g., Du et al., 2019; Xiong et al., 2020). GELU and Swish are approximately 1-Lipschitz so modern activation functions are also captured here.

Remark 38 (Implications of Theorem 36). This result establishes that:

1. Under standard architectural assumptions, convergence control requires weight manipulation
2. Any such manipulation is subject to the layer injection attack
3. Therefore, convergence-rate-based resistance methods do not provide security against adaptive attackers

This impossibility result is constructive: it precisely quantifies the attack cost as $(k-1) \cdot n$ layers for k-th root reduction. While this creates practical friction, it does not prevent a sufficiently resourced adversary.

Finally in practice, injecting a large number of linear layers could result in gradient explosion via repeated unnormalized multiplications during backpropagation. We only observed this in practice we performing the layer injection on all layers of a fully reparameterized `SmolLM2-360M-Instruct`.

J. Analysis of Second-Order Methods

Second-order optimization and curvature-aware methods are increasingly popular and feasible for large deep networks (Liu et al., 2023; Yao et al., 2021). At first glance, it may appear that approximate second-order methods could easily undo spectral reparameterization, since these methods provide preconditioners that mitigate sharp curvature. We show, however, that this intuition fails: while preconditioning can neutralize large L_1-Lipschitz constants (i.e., large Hessian eigenvalues), spectral reparameterization controls the L_2-Lipschitz constant of the Hessian itself (under the assumptions discussed in Theorem 54), which bounds the iteration complexity of second-order methods independently of preconditioning. Specifically, even exact Newton's method—which perfectly inverts the Hessian—has convergence rates that depend on L_2, and our reparameterization often inflates this quantity.

We validate this empirically in Section J.1 on a toy problem where the full Hessian is tractable. That analysis visualizes how the L_2 inflation appears concretely as large off-diagonal curvature, confirming that strong second-order optimizers—including KFAC, Natural Gradients, AdaHessian, and Cubic Newton—remain unable to circumvent spectral deformation. Large-scale second-order experiments on foundation models appear in Table 7.

For the theoretical work, a roadmap of the analysis is as follows. After defining the L_2 Hessian Lipschitz constant, we restate the convergence rate and iteration complexity of Newton's method, including full proofs for completeness. We then provide tensor-based extensions of singular value bounds to show lower bound inequalities that relate the L_2 constant to properties of the weights alone, stating the conditions under which spectral reparameterization controls the convergence of second-order methods.

J.1. Empirical Analysis

J.1.1. Demonstrating Second-Order Convergence Control

We empirically demonstrate that Spectral Deformation (SpecDef) cannot be undone by a wide range of second-order and curvature-aware optimizers on a controlled toy problem. The task is binary classification of points sampled from two intertwined spirals, trained using the three-layer ReLU MLP introduced earlier. This setting allows us to compute the full Hessian explicitly and isolate the mechanism by which SpecDef controls convergence.

Our analysis has two goals: (i) to show that SpecDef concentrates curvature primarily in off-diagonal Hessian structure,

explaining the failure of diagonal and block-diagonal approximations; and (ii) to empirically validate Theorem 3 by showing that SpecDef introduces extremely large-magnitude Hessian eigenvalues.

Experimental Setup We evaluate 200 data points across 5 random seeds, training for 1000 epochs per run. The MLP has input dimension 2 and hidden widths of 8. Unless otherwise stated, SpecDef is applied with $\alpha = 10\text{k}$. Because second-order methods are highly sensitive to hyperparameters, we conduct a full grid search for each optimizer, summarized in Table 26. We perform this search twice: once on the base model and once after applying SpecDef with $\alpha = 10^5$, ensuring a fair comparison. The resulting optimal hyperparameters are reported in Table 27.

Table 26. Hyperparameter grids used for second-order optimizer ablations.

Optimizer	Hyperparameter	Values
Adam	Learning rate	$\{10^{-9}, \ldots, 10^{-1}\}$
KFAC	Learning rate	$\{10^{-9}, \ldots, 10^{-2}\}$
	Damping	$\{10^{-3}, 10^{-2}, 3\times10^{-2}, 10^{-1}, 3\times10^{-1}\}$
Natural Gradient	Learning rate	$\{10^{-9}, \ldots, 10^{-2}\}$
	Damping	$\{0.1, 0.3, 1.0, 3.0\}$
Newton	Learning rate	$\{10^{-9}, \ldots, 10^{-2}\}$
	Damping	$\{0.1, 0.3, 1.0, 3.0\}$
AdaHessian	Learning rate	$\{10^{-9}, \ldots, 10^{-1}\}$
Cubic Newton	Learning rate	$\{10^{-9}, \ldots, 10^{-1}\}$
	σ	$\{0.1, 0.5, 1.0, 5.0\}$

SpecDef Forces Universally Smaller Learning Rates Across all optimizers, SpecDef sharply reduces the stable learning rate—often by 4–6 orders of magnitude—and increases the required damping. This empirically confirms that SpecDef enforces convergence control even for curvature-aware methods by constructing a sharply conditioned landscape that must be heavily regularized.

Table 27. Optimal optimizer hyperparameters before and after SpecDef.

Optimizer	Before SpecDef	After SpecDef
Adam	$\text{lr} = 10^{-2}$	$\text{lr} = 10^{-7}$
KFAC	$\text{lr} = 10^{-3},\ \text{damping} = 10^{-3}$	$\text{lr} = 10^{-4},\ \text{damping} = 10^{-2}$
Natural Gradient	$\text{lr} = 10^{-2},\ \text{damping} = 0.1$	$\text{lr} = 10^{-8},\ \text{damping} = 3.0$
Newton	$\text{lr} = 10^{-2},\ \text{damping} = 0.1$	$\text{lr} = 10^{-6},\ \text{damping} = 1.0$
AdaHessian	$\text{lr} = 10^{-3}$	$\text{lr} = 10^{-4}$
Cubic Newton	$\text{lr} = 0.1,\ \sigma = 5.0$	$\text{lr} = 10^{-4},\ \sigma = 5.0$

Second-order Methods Fail under SpecDef Using these tuned hyperparameters, Table 28 reports classification accuracy before and after SpecDef. After deformation, no optimizer is effective. Natural Gradient and Newton's method typically converge to poor local minima in either cases, achieving accuracies near chance (≈ 0.5), consistent with known failures of exact second-order methods in non-convex landscapes. Methods that perform well prior to SpecDef—including KFAC, AdaHessian, and Cubic Newton—are likewise rendered ineffective.

Notably, Adam performs best overall despite not being explicitly curvature-aware (though it still applies a diagonal preconditioner via second-moment estimates). This suggests that even scalable second-order approximations are insufficient to counteract SpecDef, corroborating our large-scale findings in Table 7.

The learning dynamics in Figure 10 make this effect explicit: SpecDef shifts the stable learning-rate regime downward, yielding second-order convergence control.

Table 28. Classification accuracy before and after SpecDef (mean ± std).

Optimizer	Before SpecDef	After SpecDef
KFAC	0.835 ± 0.031	0.616 ± 0.058
Natural Gradient	0.624 ± 0.005	0.589 ± 0.045
Newton	0.625 ± 0.006	0.621 ± 0.007
AdaHessian	0.795 ± 0.048	0.648 ± 0.027
Cubic Newton	0.748 ± 0.038	0.642 ± 0.026
Adam	0.806 ± 0.042	0.648 ± 0.011

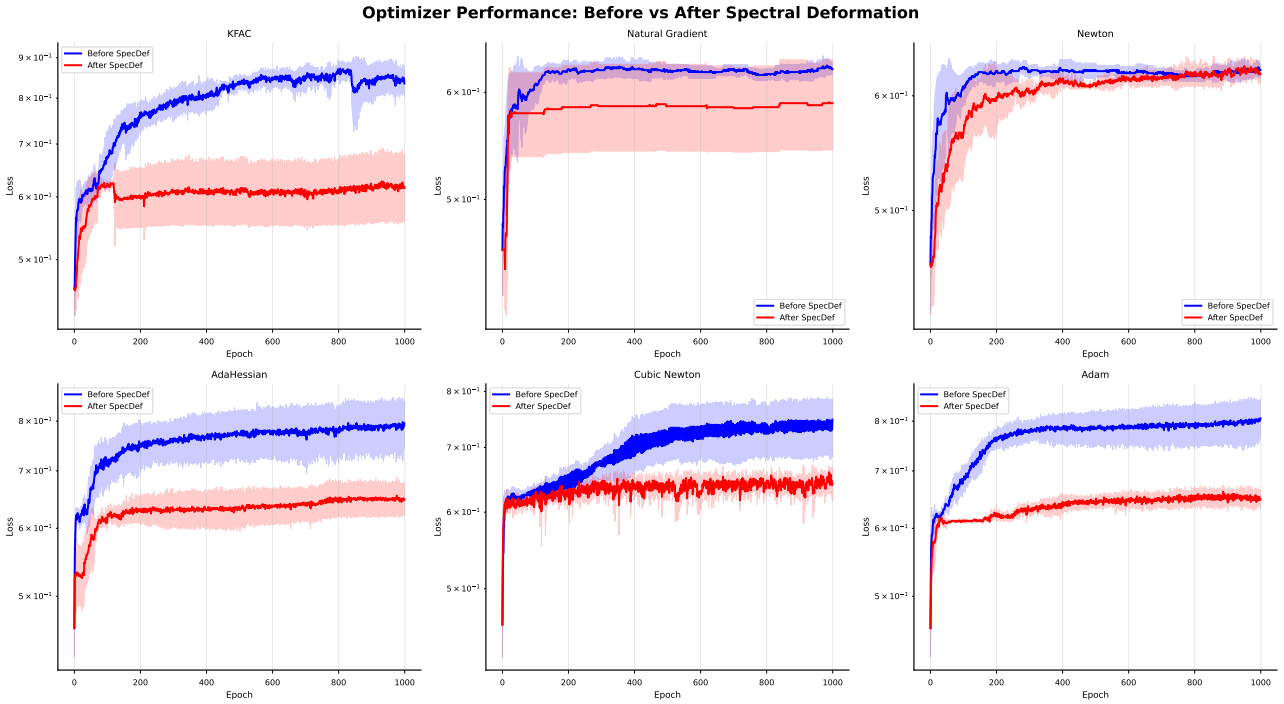


Figure 10. SpecDef sharply contracts the stable learning-rate regime, demonstrating effective second-order convergence control.

J.1.2. Hessian Analysis

To explain why SpecDef remains effective even against second-order optimization, we analyze the full Hessian before and after deformation. Figure 11 shows both the Hessian heatmap and its eigenvalue spectrum.

Before SpecDef, the spectrum follows a distribution consistent with random matrix theory (Baskerville et al., 2022a). After SpecDef (with $k = 4$), the spectrum spreads across many orders of magnitude: most curvature directions remain relatively flat, but a small number become extremely sharp, both positively and negatively. This heavy-tailed spectrum directly instantiates the bound in Theorem 3.

Crucially, the dominant curvature after SpecDef concentrates in off-diagonal Hessian blocks. This explains the empirical failure of diagonal and block-diagonal curvature approximations: the sharpest directions are in places these methods do not approximate.

We further vary both the number of modified singular values and the choice of reparameterized layers. As shown in Figure 12, increasing the number of modified singular values systematically sharpens the landscape, while changing which layers are deformed redistributes where curvature concentrates. These patterns provide a second, independent empirical validation that manipulating weight spectra alone suffices to control second-order curvature.

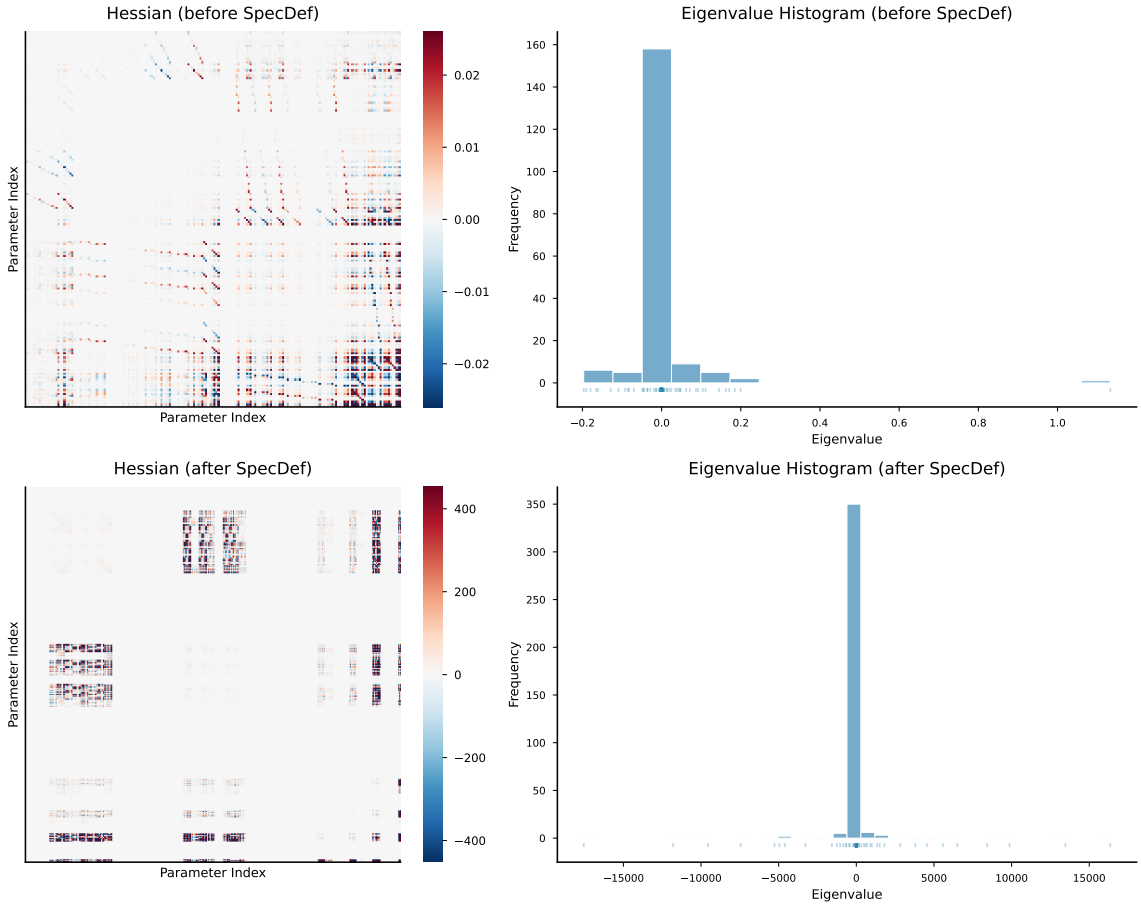


Figure 11. Hessian heatmaps and eigenvalue distributions before and after SpecDef. SpecDef's most pronounced effect is in the off-diagonals.

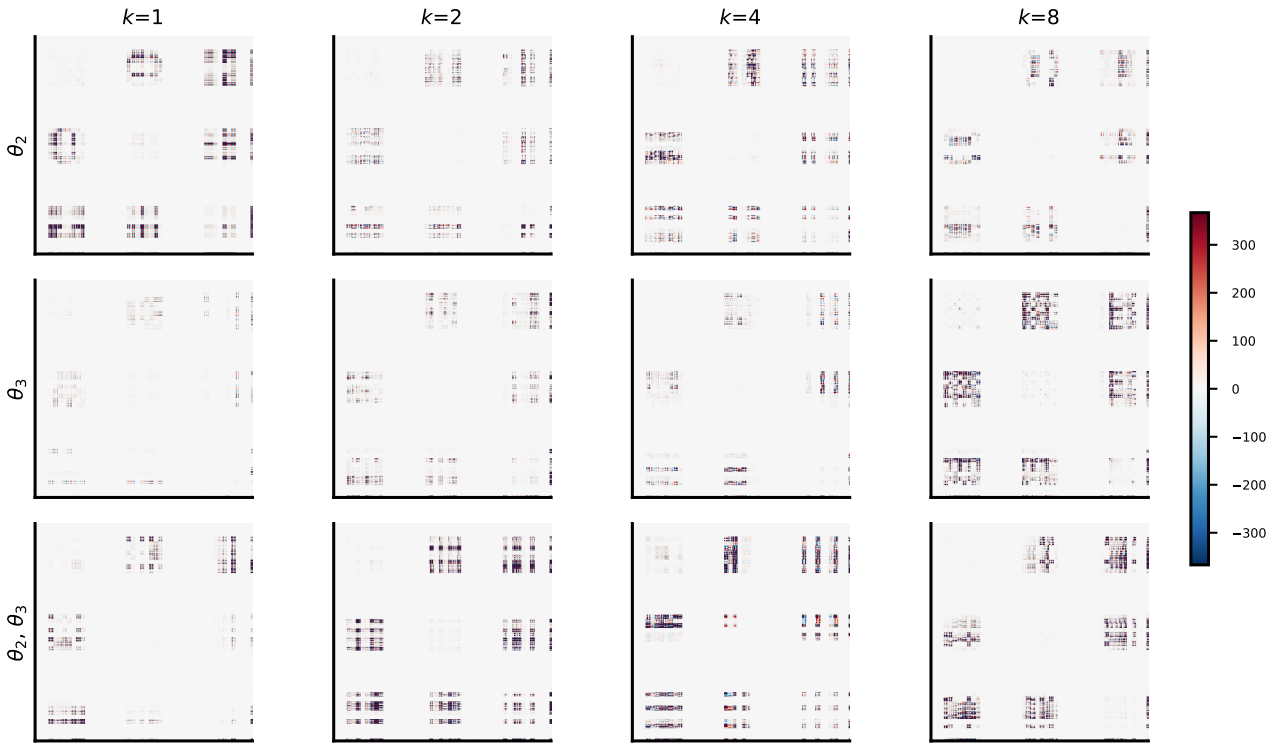


Figure 12. Hessian heatmaps as a function of the number and location of modified singular values. The pattern is consistent regardless of selected k or layer.

J.2. Theoretical Analysis of Muon

In order to explain why Muon (Jordan et al., 2024) fails to converge given a sufficient α used for SpecDef, we briefly review the Muon optimizer and its known convergence rate. We will use the notation from Li & Hong (2025) rather than our paper to better connect our results with theirs.

Definition 39 (Muon optimizer in heavy-ball form)**.** Following Li & Hong (2025), we consider minimizing an objective function $f(X)$ over $X \in \mathbb{R}^{m\times n}$ where X represents a parameter matrix. At iteration t, Muon samples a mini-batch $\{\xi_{t,i}\}_{i=1}^{B}$ and forms the stochastic gradient

$$G_t \;\leftarrow\; \frac{1}{B}\sum_{i=1}^{B}\nabla f(X_t,\xi_{t,i}) \in \mathbb{R}^{m\times n}. \tag{5}$$

Muon maintains a momentum matrix (heavy-ball style)

$$B_t \;\leftarrow\; \beta B_{t-1} + (1-\beta)G_t, \tag{6}$$

where $\beta \in (0,1)$.

Let $B_t = U_t S_t V_t^\top$ be an SVD result (with $U_t \in \mathbb{R}^{m\times r}$, $S_t \in \mathbb{R}^{r\times r}$, $V_t \in \mathbb{R}^{n\times r}$, and $r = \operatorname{rank}(B_t)$). Muon discards singular values and uses only the singular directions:

$$O_t \;\leftarrow\; U_t V_t^\top, \tag{7}$$

then updates

$$X_{t+1} \;\leftarrow\; X_t - \eta_t O_t, \tag{8}$$

where $\eta_t > 0$ is the step size. (Compared to a Nesterov-type presentation, we use the above heavy-ball form where G_t is explicitly multiplied by $(1-\beta)$ in the momentum update.)

Theorem 40 (Convergence of Muon optimizer)**.** *Assume f is L-smooth in Frobenius norm, i.e., $\|\nabla f(X) - \nabla f(Y)\|_F \le L\|X-Y\|_F$, and the stochastic gradient is unbiased with bounded variance: $\mathbb{E}[\nabla f(X,\xi)] = \nabla f(X)$ and $\mathbb{E}\|\nabla f(X,\xi) - \nabla f(X)\|_F^2 \le \sigma^2$. Let $R = f(X_1) - f^*$, where $f^* = \inf_X f(X)$ denotes the optimal objective value. All expectations are taken with respect to the stochasticity of the mini-batch sampling.*

Take $\beta = 1-\alpha$ with $\alpha = \min\!\left(\frac{\sqrt{RL}}{\sigma\sqrt{T}}, 1\right)$, use a constant step size $\eta_t = \eta$ with $\eta = \sqrt{\frac{4R}{(10/\alpha+2n)\,TL}}$, and set $B = 1$. Then the Muon iterates in Definition 39 satisfy

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\nabla f(X_t)\|_F\big] \;\le\; \mathcal{O}\!\left(\frac{\sqrt{nRL}}{\sqrt{T}} \;+\; \frac{\sigma^2}{\sqrt{RLT}} \;+\; \frac{\sqrt{\sigma}\,(RL)^{1/4}}{T^{1/4}}\right). \tag{9}$$

If β is an arbitrary fixed constant, take $B = T$. Then one obtains

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(X_t)\|_F \;\le\; \mathcal{O}\!\left(\frac{\sqrt{nRL}}{\sqrt{T}} \;+\; \frac{\sigma}{T^{3/2}} \;+\; \frac{\sigma}{\sqrt{T}}\right). \tag{10}$$

Proof. *The proof follows directly from the analysis in Li & Hong (2025).*

At first glance, one might expect Muon to perform well as the model spectra is increased or deformed, since its SVD-based update removes singular value magnitudes and appears robust to ill-conditioning. However, the convergence rate reveals a more fundamental limitation.

Specifically, the leading term in the convergence bound scales as $\mathcal{O}(\sqrt{nRL/T})$, where L denotes the Lipschitz smoothness constant of the objective, capturing the overall curvature of the loss landscape. Increasing the SpecDef strength α applies stronger spectral deformation to the model parameters, which in turn increases the effective curvature of the objective function and thus increases L Theorem 3. As a result, the step size required for stable optimization becomes smaller, and convergence slows down under a fixed number of training iterations. This effect cannot be addressed by the SVD-based update used in Muon, which normalizes the update direction but does not change the global curvature of the objective.

This behaviour is reflected in Table 7. As α increases, Muon requires more iterations and achieves lower success rates, consistent with the curvature-dependent limitation predicted by the convergence analysis.

J.3. Theoretical Analysis of Second-Order Methods

In this section we will show how L_2 relates to the spectral values of the weight matrices of neural networks θ_i. We will begin with some preliminary definitions necessary for understanding the iteration complexity of Newton's method which will be our canonical second order method we use for analysis. We will then employ a tensor analysis specifically using the T-SVD of Zhang et al. (2021) to show a number of analagous results to our first-order ones: an L_2 bound in $\nabla^3_\theta f$, versions of the matrix singular value inequalities for the tensors, and finally applications to neural networks which will explain the empirical results we obtained above.

J.3.1. Preliminaries

Definition 41 (L_2-Lipschitz Continuous Hessian)**.** A twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to have an *L_2-Lipschitz continuous Hessian* if there exists a constant $L_2 > 0$ such that for all $x, y \in \mathbb{R}^n$,

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \le L_2 \|x - y\|.$$

in which case we call L_2 the *Hessian Lipschitz constant*.

Theorem 42 (Jensen's Spectral Norm Inequality)**.** *Let $S(u) : [0, 1] \to \mathbb{R}^{n\times n}$ be a integrable matrix-valued function. Then its spectral norm satisfies*

$$\Big\| \int_0^1 S(u)\, du \Big\| \le \int_0^1 \|S(u)\|\, du.$$

This result is a matrix version of Jensen's inequality for the spectral norm. A detailed proof can be found in Zhang (2023).

Lemma 43 (Lipschitz Hessian Bound)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function with an L_2-Lipschitz continuous Hessian as defined in Definition 41. Then, for any two points $x, y \in \mathbb{R}^n$, the corresponding Hessians satisfy*

$$\nabla^2 f(x) - L_2\|x - y\|\, I \;\preceq\; \nabla^2 f(y) \;\preceq\; \nabla^2 f(x) + L_2\|x - y\|\, I.$$

This lemma provides upper and lower bounds on the variation of the Hessian under the Lipschitz continuity assumption, quantifying how much the curvature may change between two nearby points. A detailed proof is given in Zhang (2023).

J.3.2. Convergence Rate of Newton's Method on Lipschitz-Hessian Functions

We now state the convergence rate of Newton's method and it's iteration complexity to parallel our analysis above with first-order methods.

Theorem 44 (Local quadratic convergence of Newton's method)**.** *Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function, and suppose $\nabla\mathcal{L}(\theta_*) = 0$ and $\nabla^2\mathcal{L}(\theta_*) \succeq \mu I$ for some $\mu > 0$, where μ denotes the local eigenvalue lower bound of the Hessian at θ_*. Assume further that $\mathcal{L}$ has an L_2-Lipschitz continuous Hessian as defined in Definition 41.*

Then, for any constant $\rho \in (0, \mu/L_2)$, if an iterate satisfies $\|\theta_k - \theta_*\| \le \rho$, the Newton update

$$\theta_{k+1} = \theta_k - [\nabla^2\mathcal{L}(\theta_k)]^{-1}\nabla\mathcal{L}(\theta_k)$$

satisfies the local quadratic convergence bound:

$$\|\theta_{k+1} - \theta_*\| \;\le\; \psi(\rho)\, \|\theta_k - \theta_*\|^2, \qquad \psi(\rho) := \frac{L_2}{2(\mu - L_2\rho)}.$$

Furthermore, if $\psi(\rho)\rho < 1$ (equivalently, $\rho < 2\mu/(3L_2)$), then the neighborhood $\|\theta - \theta_*\| \le \rho$ is strictly invariant and the iteration is contractive:

$$\|\theta_{k+1} - \theta_*\| < \|\theta_k - \theta_*\|.$$

Under this condition, Newton's method converges quadratically with a monotonically decreasing error norm.

Proof. Consider the straight-line path $\Gamma : \theta(u) = \theta_* + u(\theta_k - \theta_*)$, for $u \in [0, 1]$. By the Fundamental Theorem of Calculus, we obtain:

$$\nabla\mathcal{L}(\theta_k) - \nabla\mathcal{L}(\theta_*) = \int_\Gamma \nabla^2\mathcal{L}(\theta)\, d\theta = \int_0^1 \nabla^2\mathcal{L}(\theta(u))\frac{d\theta}{du}\, du = \int_0^1 \nabla^2\mathcal{L}\big(\theta_* + u(\theta_k - \theta_*)\big)(\theta_k - \theta_*)\, du. \tag{11}$$

Similarly, $\theta_k - \theta_*$ can be expressed as:

$$\theta_k - \theta_* = [\nabla^2 \mathcal{L}(\theta_k)]^{-1} \int_0^1 \nabla^2 \mathcal{L}(\theta_k)(\theta_k - \theta_*)\, du. \tag{12}$$

From the Newton update rule:

$$\theta_{k+1} - \theta_* = \theta_k - \theta_* - [\nabla^2 \mathcal{L}(\theta_k)]^{-1} \big[\nabla \mathcal{L}(\theta_k) - \nabla \mathcal{L}(\theta_*)\big].$$

Substituting (11) and (12) yields:

$$\theta_{k+1} - \theta_* = [\nabla^2 \mathcal{L}(\theta_k)]^{-1} \mathcal{R}_k(\theta_k - \theta_*), \quad \mathcal{R}_k = \int_0^1 [\nabla^2 \mathcal{L}(\theta_k) - \nabla^2 \mathcal{L}(\theta_* + u(\theta_k - \theta_*))]\, du.$$

By Definition 41 and Theorem 42, there exists $L_2 > 0$ such that

$$\|\mathcal{R}_k\| \le \int_0^1 \|\nabla^2 \mathcal{L}(\theta_k) - \nabla^2 \mathcal{L}(\theta_* + u(\theta_k - \theta_*))\|\, du \le \int_0^1 L_2(1-u)\|\theta_k - \theta_*\|\, du = \tfrac{1}{2} L_2 \|\theta_k - \theta_*\|.$$

Furthermore, applying Lemma 43 with $x = \theta_*$ and $y = \theta_k$, we obtain:

$$\nabla^2 \mathcal{L}(\theta_*) - L_2\|\theta_k - \theta_*\|\, I \;\preceq\; \nabla^2 \mathcal{L}(\theta_k) \;\preceq\; \nabla^2 \mathcal{L}(\theta_*) + L_2\|\theta_k - \theta_*\|\, I.$$

Since $\nabla^2 \mathcal{L}(\theta_*) \succeq \mu I$, it follows that

$$\nabla^2 \mathcal{L}(\theta_k) \;\succeq\; \big(\mu - L_2\|\theta_k - \theta_*\|\big) I.$$

Thus, when $\|\theta_k - \theta_*\| < \mu / L_2$:

$$\big\|[\nabla^2 \mathcal{L}(\theta_k)]^{-1}\big\| \;\le\; \frac{1}{\mu - L_2\|\theta_k - \theta_*\|} \quad .$$

Taking norms in the relation $\theta_{k+1} - \theta_* = [\nabla^2 \mathcal{L}(\theta_k)]^{-1} \mathcal{R}_k(\theta_k - \theta_*)$ yields

$$\|\theta_{k+1} - \theta_*\| \le \big\|[\nabla^2 \mathcal{L}(\theta_k)]^{-1}\big\|\, \|\mathcal{R}_k\|\, \|\theta_k - \theta_*\| \le \frac{L_2}{2\big(\mu - L_2\|\theta_k - \theta_*\|\big)}\, \|\theta_k - \theta_*\|^2.$$

Note that the bound $\|\theta_k - \theta_*\| < \mu / L_2$ ensures that the Hessian remains positive definite and invertible within the local neighborhood of θ_*, since $\nabla^2 \mathcal{L}(\theta_k) \succeq (\mu - L_2\|\theta_k - \theta_*\|) I \succ 0$. To formalize this region of guaranteed invertibility, we fix a constant $\rho \in (0, \mu/L_2)$ and suppose $\|\theta_k - \theta_*\| \le \rho$, then:

$$\|\theta_{k+1} - \theta_*\| \;\le\; \frac{L_2}{2(\mu - L_2\rho)}\, \|\theta_k - \theta_*\|^2 \;=\; \psi(\rho)\, \|\theta_k - \theta_*\|^2, \qquad \psi(\rho) := \frac{L_2}{2(\mu - L_2\rho)}.$$

This is the stated local quadratic bound. Moreover, if $\psi(\rho)\rho < 1$, then:

$$\|\theta_{k+1} - \theta_*\| \;\le\; \psi(\rho)\, \rho^2 \;<\; \rho \quad \text{and} \quad \frac{\|\theta_{k+1} - \theta_*\|}{\|\theta_k - \theta_*\|} \;\le\; \psi(\rho)\, \|\theta_k - \theta_*\| \;\le\; \psi(\rho)\, \rho \;<\; 1,$$

which shows that the neighborhood $\{\theta : \|\theta - \theta_*\| \le \rho\}$ is strictly invariant and the iteration is contractive with a monotonically decreasing error norm. □

This theorem demonstrates that Newton's method achieves local quadratic convergence when the objective function has a strongly positive definite Hessian at the optimum and the Hessian varies smoothly in a neighborhood of that point. This proof is a restatement and detailed derivation of Theorem 3.3 in Zhang (2023).

J.3.3. Iteration Complexity of Newton's Method

From Theorem 44, the local quadratic convergence relation is given by

$$e_{k+1} \le \psi(\rho)\, e_k^2, \qquad \psi(\rho) = \frac{L_2}{2(\mu - L_2\rho)},$$

where $e_k = \|\theta_k - \theta_*\|$ and $\rho < \mu/L_2$ ensures that the Hessian remains positive definite and the iteration stays within the local invariant region.

We now derive the corresponding *iteration complexity*, that is, the number of steps T required to achieve $e_T \le \varepsilon$ for a given accuracy $\varepsilon > 0$.

Unrolling the recurrence relation gives

$$e_k \le \psi(\rho) e_{k-1}^2 \le \psi(\rho)^{2^0+2^1} e_{k-2}^{2^2} \le \cdots \le \psi(\rho)^{\sum_{i=0}^{k-1} 2^i} e_0^{2^k} = \psi(\rho)^{2^k-1} e_0^{2^k}.$$

To determine T such that $e_T \le \varepsilon$, we require

$$\psi(\rho)^{2^T-1} e_0^{2^T} \le \varepsilon \quad \Leftrightarrow \quad \big(\psi(\rho)e_0\big)^{2^T} \le \psi(\rho)\varepsilon.$$

Taking natural logarithms on both sides, and noting that by the local quadratic convergence condition we already have $e_1 = \psi(\rho)e_0^2 < e_0$ (hence $\psi(\rho)e_0 < 1$), we obtain

$$2^T \ln\big(\psi(\rho)e_0\big) \le \ln\big(\psi(\rho)\varepsilon\big).$$

Since $\ln(\psi(\rho)e_0) < 0$, dividing both sides by $\ln(\psi(\rho)e_0)$ reverses the inequality, giving

$$2^T \;\ge\; \frac{\ln\big(\psi(\rho)\varepsilon\big)}{\ln\big(\psi(\rho)e_0\big)} \;=\; \frac{\ln\big(1/(\psi(\rho)\varepsilon)\big)}{\ln\big(1/(\psi(\rho)e_0)\big)}.$$

Finally, taking $\log_2$ on both sides (monotonicity of $\log_2$) yields

$$T \;\ge\; \log_2\left(\frac{\ln\big(1/(\psi(\rho)\varepsilon)\big)}{\ln\big(1/(\psi(\rho)e_0)\big)}\right) \;=\; \log_2\left(\frac{\ln\big(1/(\psi(\rho)\varepsilon)\big)}{\ln\big(1/(\psi(\rho)\|\theta_0 - \theta_*\|)\big)}\right).$$

This gives an explicit lower bound on the number of iterations required for Newton's method to reach a target accuracy ε within the local quadratic convergence region and since $\psi(\rho) = \frac{L_2}{2(\mu - L_2\rho)}$ there is a clear dependency of the iteration complexity on L_2.

J.3.4. Third Order Analysis: Preliminaries

Our first goal to parallel the first-order analysis in the paper is to extend Theorem 13 to the third-order derivative tensor $\mathcal{T}(x) := \nabla^3 f(x) \in \mathbb{R}^{n\times n\times n}$, which represents the derivative of the Hessian of a thrice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$.

Since we work within the T-product and T-SVD framework from Zhang et al. (2021), we will present a number of preliminary definitions below.

Definition 45 (Unfold, Fold, and Block-Circulant Representation)**.** Let $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ be a third-order tensor with frontal slices $A_k = \mathcal{A}(:,:,k) \in \mathbb{R}^{m\times n}$ for $k = 1, \ldots, p$. The *unfolding* of $\mathcal{A}$ is defined as the block column concatenation

$$\mathrm{unfold}(\mathcal{A}) = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{bmatrix} \in \mathbb{R}^{mp\times n},$$

The *folding* operator $\mathrm{fold}(\cdot)$ is the inverse mapping that reconstructs a tensor from its unfolded matrix:

$$\mathrm{fold}(\mathrm{unfold}(\mathcal{A})) = \mathcal{A}.$$

The *block-circulant matrix* of $\mathcal{A}$, denoted by $\mathrm{bcirc}(\mathcal{A}) \in \mathbb{R}^{mp\times np}$, is defined as

$$\mathrm{bcirc}(\mathcal{A}) = \begin{bmatrix} A_1 & A_p & A_{p-1} & \cdots & A_2 \\ A_2 & A_1 & A_p & \cdots & A_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A_p & A_{p-1} & \cdots & A_2 & A_1 \end{bmatrix}.$$

The block-circulant structure is fundamental to the T-product framework, as it enables the extension of matrix algebraic operations to third-order tensors through the use of discrete Fourier transforms.

Definition 46 (Discrete Fourier Transform along the third dimension)**.** For a tensor $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$, the Discrete Fourier Transform (DFT) along the third dimension is defined as

$$\widehat{\mathcal{A}}(:,:,k) = \sum_{t=1}^{p} \mathcal{A}(:,:,t)\, e^{-2\pi i(t-1)(k-1)/p}, \quad k = 1,\dots,p.$$

The inverse DFT is given by

$$\mathcal{A}(:,:,t) = \frac{1}{p}\sum_{k=1}^{p} \widehat{\mathcal{A}}(:,:,k)\, e^{2\pi i(t-1)(k-1)/p}.$$

Under this transform, the block-circulant matrix of $\mathcal{A}$ is block-diagonalized as

$$(F_p \otimes I_m)\,\mathrm{bcirc}(\mathcal{A})\,(F_p^{-1}\otimes I_n) = \mathrm{diag}\big(\widehat{\mathcal{A}}(:,:,1),\dots,\widehat{\mathcal{A}}(:,:,p)\big),$$

where F_p is the $p\times p$ DFT matrix with entries $[F_p]_{jk} = e^{-2\pi i(j-1)(k-1)/p}$, and $\otimes$ denotes the Kronecker product.

In practice, the DFT and inverse DFT are efficiently computed using the *Fast Fourier Transform (FFT)* and *Inverse Fast Fourier Transform (IFFT)*. For simplicity, we use the notations $\widehat{\mathcal{A}} = \mathrm{fft}(\mathcal{A}, [], 3)$ and $\mathcal{A} = \mathrm{ifft}(\widehat{\mathcal{A}}, [], 3)$ throughout the paper (Zhang et al., 2021).

Throughout this appendix, we adopt the *unitary* version of the above DFT, i.e., we normalize the DFT matrix F_p by $1/\sqrt{p}$ so that F_p is unitary and the associated Fourier basis vectors have unit ℓ_2 norm. With this convention, all block-diagonalization identities remain valid, and tensor singular values are unchanged up to a consistent scaling.

Definition 47 (Tensor Product (T-product))**.** Let $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ and $\mathcal{B} \in \mathbb{R}^{n\times t\times p}$. The *T-product* between $\mathcal{A}$ and $\mathcal{B}$ is defined by

$$\mathcal{A} * \mathcal{B} = \mathrm{fold}\big(\mathrm{bcirc}(\mathcal{A})\,\mathrm{unfold}(\mathcal{B})\big) \in \mathbb{R}^{m\times t\times p}.$$

According to Definition 46, if $\widehat{\mathcal{A}} = \mathrm{fft}(\mathcal{A}, [], 3)$ and $\widehat{\mathcal{B}} = \mathrm{fft}(\mathcal{B}, [], 3)$, then

$$\widehat{\mathcal{C}}(:,:,k) = \widehat{\mathcal{A}}(:,:,k)\,\widehat{\mathcal{B}}(:,:,k), \quad \forall k = 1,\dots,p,$$

where $\widehat{\mathcal{C}} = \mathrm{fft}(\mathcal{A} * \mathcal{B}, [], 3)$.

This definition follows Zhang et al. (2021) and establishes the foundation for tensor algebra operations, allowing tensors to perform matrix-like multiplication in the Fourier domain that can be efficiently computed using FFT.

We will now present a singular value decomposition for tensors known as the T-SVD.

Theorem 48 (Tensor Singular Value Decomposition (T-SVD) and Tensor Singular Values)**.** *Let $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ be a third-order tensor. Then $\mathcal{A}$ admits the following factorization under the T-product:*

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^\top,$$

where $\mathcal{U} \in \mathbb{R}^{m\times m\times p}$ and $\mathcal{V} \in \mathbb{R}^{n\times n\times p}$ are orthogonal tensors satisfying $\mathcal{U}^\top * \mathcal{U} = \mathcal{I}$ and $\mathcal{V}^\top * \mathcal{V} = \mathcal{I}$, and $\mathcal{S} \in \mathbb{R}^{m\times n\times p}$ is an f-diagonal tensor*, that is, each frontal slice $\mathcal{S}(:,:,k)$ is diagonal. According to Definition 46, if $\widehat{\mathcal{A}} = \mathrm{fft}(\mathcal{A}, [], 3)$, then each frontal slice of $\widehat{\mathcal{A}}$ admits a standard matrix SVD:*

$$\widehat{\mathcal{A}}(:,:,k) = \widehat{U}_k\,\widehat{S}_k\,\widehat{V}_k^\top, \quad k = 1,\dots,p,$$

where $\widehat{U}_k, \widehat{V}_k$ are orthogonal matrices and $\widehat{S}_k = \operatorname{diag}\big(\sigma_1^{(k)}, \ldots, \sigma_r^{(k)}\big)$ with $r = \min(m, n)$. Applying the inverse transform defined in Definition 46 yields

$$\mathcal{U} = \operatorname{ifft}(\widehat{\mathcal{U}}, [], 3), \quad \mathcal{S} = \operatorname{ifft}(\widehat{\mathcal{S}}, [], 3), \quad \mathcal{V} = \operatorname{ifft}(\widehat{\mathcal{V}}, [], 3).$$

The **tensor singular values of $\mathcal{A}$ are defined as the diagonal entries of the first frontal slice $\mathcal{S}(:, :, 1)$, denoted by**

$$\varsigma_i(\mathcal{A}) := \mathcal{S}(i, i, 1) = \frac{1}{p} \sum_{k=1}^{p} \sigma_i(\widehat{\mathcal{A}}(:, :, k)), \quad i = 1, \ldots, r,$$

Hence, the tensor singular values satisfy the monotonicity

$$\varsigma_1(\mathcal{A}) \geq \varsigma_2(\mathcal{A}) \geq \cdots \geq \varsigma_r(\mathcal{A}) \geq 0.$$

This theorem extends the classical matrix SVD to third-order tensors through the T-product framework. See the proof in Theorem 2.3 from Zhang et al. (2021).

J.3.5. Third Order Analysis: L_2 bounds

We not present a tensor Version of Theorem 13 where for the tensor singular values ς_i, we have $\varsigma_1\big(\mathcal{T}(x)\big) \leq L_2$.

Theorem 49 (Tensor Version of Theorem 13)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be thrice continuously differentiable with an L_2-Lipschitz continuous Hessian as defined in Definition 41. Let $\mathcal{T}(x) = \nabla^3 f(x) \in \mathbb{R}^{n \times n \times n}$ denote the third-order derivative tensor. Then the largest tensor singular value of $\mathcal{T}(x)$ under the T–SVD framework satisfies*

$$\varsigma_1\big(\mathcal{T}(x)\big) \leq L_2.$$

Proof. Let f satisfy the L_2-Lipschitz Hessian condition in Definition 41, namely

$$\|\nabla^2 f(u) - \nabla^2 f(v)\| \leq L_2 \|u - v\|, \qquad \forall\, u, v \in \mathbb{R}^n.$$

Fix a point $x \in \mathbb{R}^n$ and a unit direction $h \in \mathbb{R}^n$ with $\|h\|_2 = 1$. Define the matrix-valued function

$$G(t) := \nabla^2 f(x + th), \qquad t \in [0, 1].$$

Then by Definition 41,

$$\|G(t) - G(s)\| = \|\nabla^2 f(x + th) - \nabla^2 f(x + sh)\| \leq L_2 |t - s|.$$

Hence G is an L_2-Lipschitz function of t. Since f is C^3 and G is differentiable then

$$G'(t) = \frac{d}{dt} \nabla^2 f(x + th) = \nabla^3 f(x + th)[\,\cdot, \cdot, h\,].$$

Since G is differentiable and its derivative $G'(t)$ is continuous, for each fixed t we have

$$\|G'(t)\| = \Big\| \lim_{s \to t} \frac{G(t) - G(s)}{t - s} \Big\| \leq \limsup_{s \to t} \frac{\|G(t) - G(s)\|}{|t - s|} \leq L_2, \qquad \forall\, t \in [0, 1].$$

Therefore,

$$\big\| \nabla^3 f(x + th)[\,\cdot, \cdot, h\,] \big\|_2 = \|G'(t)\| \leq L_2, \qquad \forall\, t \in [0, 1],\ \|h\|_2 = 1. \tag{13}$$

Applying the discrete Fourier transform along the third tensor dimension (Definition 46) gives

$$\widehat{\mathcal{T}} = \operatorname{fft}(\mathcal{T}(x), [], 3), \qquad \mathcal{T}(x) = \nabla^3 f(x).$$

Its k-th frontal slice satisfies

$$\widehat{\mathcal{T}}(:, :, k) = \mathcal{T}(x)[\,\cdot, \cdot, \phi_k\,],$$

where ϕ_k denotes the k-th unit-norm Fourier basis vector under the unitary DFT convention adopted throughout this appendix. Moreover, the bound in (13) extends to complex unit vectors by identifying $\mathbb{C}^n$ with $\mathbb{R}^{2n}$ and using the same operator norm.

Since $\|\phi_k\|_2 = 1$, substituting $h = \phi_k$ into (13) yields

$$\sigma_1\big(\widehat{\mathcal{T}}(:,:,k)\big) = \big\|\widehat{\mathcal{T}}(:,:,k)\big\|_2 \le L_2, \qquad k = 1, \dots, p.$$

From Theorem 48, the largest tensor singular values of $\mathcal{T}(x)$ are

$$\varsigma_1\big(\mathcal{T}(x)\big) = \frac{1}{p}\sum_{k=1}^{p} \sigma_1\big(\widehat{\mathcal{T}}(:,:,k)\big) \le \frac{1}{p}\sum_{k=1}^{p} L_2 = L_2.$$

Therefore the largest tensor singular value of the third-order derivative tensor $\nabla^3 f(x)$ is bounded above by the Hessian Lipschitz constant L_2. □

J.3.6. Third Order Analysis: Singular Value Inequalities in Tensors

Before we proceed with connecting the above L_2 bounds with increases to weight singular values as we did with L_1 previously, we will need to state some tensor singular value inequalities analogous to the matrix singular values we presented in Section B.

Theorem 50 (Poincaré separation theorem for tensor singular values)**.** *Let $\mathcal{A} \in \mathbb{R}^{m\times n\times p}$ be a third-order tensor, and let $\mathcal{A}_{\text{sub}} \in \mathbb{R}^{m'\times n'\times p}$ be any subtensor of $\mathcal{A}$ with $1 \le m' \le m$ and $1 \le n' \le n$. Let the tensor singular values (under the T–SVD in Theorem 48) be*

$$\varsigma_1(\mathcal{A}) \ge \varsigma_2(\mathcal{A}) \ge \cdots \ge \varsigma_r(\mathcal{A}) \ge 0, \quad r = \min\{m, n\},$$

and

$$\varsigma_1(\mathcal{A}_{\text{sub}}) \ge \varsigma_2(\mathcal{A}_{\text{sub}}) \ge \cdots \ge \varsigma_{r'}(\mathcal{A}_{\text{sub}}) \ge 0, \quad r' = \min\{m', n'\}.$$

Then, for all $k = 1, \dots, r'$, we have

$$\varsigma_k(\mathcal{A}) \;\ge\; \varsigma_k(\mathcal{A}_{\text{sub}}) \;\ge\; \varsigma_{k+(m-m')+(n-n')}(\mathcal{A}), \tag{14}$$

with the convention that $\varsigma_j(\mathcal{A}) = 0$ if $j > r$.

Proof. Apply the discrete Fourier transform along the third mode (Definition 46) to both $\mathcal{A}$ and $\mathcal{A}_{\text{sub}}$:

$$\widehat{\mathcal{A}} = \text{fft}(\mathcal{A}, [], 3), \qquad \widehat{\mathcal{A}_{\text{sub}}} = \text{fft}(\mathcal{A}_{\text{sub}}, [], 3).$$

For each $k = 1, \dots, p$, the k-th frontal slices $\widehat{\mathcal{A}}(:,:,k)$ and $\widehat{\mathcal{A}_{\text{sub}}}(:,:,k)$ are matrices with sizes $m \times n$ and $m' \times n'$, respectively, and $\widehat{\mathcal{A}_{\text{sub}}}(:,:,k)$ is a principal submatrix of $\widehat{\mathcal{A}}(:,:,k)$. By the classical Poincaré separation (Horn & Johnson, 1991, Th. 3.1.2), we have for each $k = 1, \dots, p$:

$$\sigma_j\big(\widehat{\mathcal{A}}(:,:,k)\big) \;\ge\; \sigma_j\big(\widehat{\mathcal{A}_{\text{sub}}}(:,:,k)\big) \;\ge\; \sigma_{\,j+(m-m')+(n-n')}\big(\widehat{\mathcal{A}}(:,:,k)\big), \qquad j = 1, \dots, r'.$$

Averaging over $k = 1, \dots, p$ and using the definition of tensor singular values in Theorem 48,

$$\varsigma_j(\mathcal{A}) = \frac{1}{p}\sum_{k=1}^{p} \sigma_j\big(\widehat{\mathcal{A}}(:,:,k)\big), \qquad \varsigma_j(\mathcal{A}_{\text{sub}}) = \frac{1}{p}\sum_{k=1}^{p} \sigma_j\big(\widehat{\mathcal{A}_{\text{sub}}}(:,:,k)\big),$$

we obtain (14). The result is thus a direct tensor analogue of the matrix Poincaré separation theorem (see also Zhang et al., 2021, Theorem 3.1). □

Theorem 51 (Additive Tensor Singular Value Bound)**.** *Let $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{m\times n\times p}$ and let $q = \min\{m, n\}$. Denote the tensor singular values under the T–SVD (Definition 48) by $\varsigma_1(\cdot) \ge \cdots \ge \varsigma_q(\cdot) \ge 0$. Then for all $i = 1, \dots, q$,*

$$\varsigma_i(\mathcal{A}) - \varsigma_1(\mathcal{B}) \;\le\; \varsigma_i(\mathcal{A} + \mathcal{B}) \;\le\; \varsigma_i(\mathcal{A}) + \varsigma_1(\mathcal{B}). \tag{15}$$

Proof. Let $\widehat{\mathcal{A}} = \mathrm{fft}(\mathcal{A}, [], 3)$ and $\widehat{\mathcal{B}} = \mathrm{fft}(\mathcal{B}, [], 3)$. By linearity of the DFT along the third mode,

$$\widehat{\mathcal{A}+\mathcal{B}}(:,:,k) = \widehat{\mathcal{A}}(:,:,k) + \widehat{\mathcal{B}}(:,:,k), \qquad k = 1, \dots, p.$$

For each slice k, applying Theorem 16 to the matrices $\widehat{A}_k$ and $\widehat{B}_k$ yields

$$\sigma_i(\widehat{A}_k) - \sigma_1(\widehat{B}_k) \;\le\; \sigma_i(\widehat{A}_k + \widehat{B}_k) \;\le\; \sigma_i(\widehat{A}_k) + \sigma_1(\widehat{B}_k), \qquad i = 1, \dots, q.$$

Equivalently,

$$\big|\sigma_i(\widehat{A}_k + \widehat{B}_k) - \sigma_i(\widehat{A}_k)\big| \;\le\; \sigma_1(\widehat{B}_k), \qquad i = 1, \dots, q.$$

Summing over $k = 1, \dots, p$, dividing by p, and using the definition of tensor singular values under the T–SVD yields

$$\big|\varsigma_i(\mathcal{A}+\mathcal{B}) - \varsigma_i(\mathcal{A})\big| \le \frac{1}{p}\sum_{k=1}^{p} \sigma_1(\widehat{\mathcal{B}}(:,:,k)) = \varsigma_1(\mathcal{B}),$$

which is equivalent to (15). □

Theorem 52 (Multiplicative Tensor Singular Value Bounds)**.** *Let $\mathcal{A} \in \mathbb{R}^{m\times\ell\times p}$ and $\mathcal{B} \in \mathbb{R}^{\ell\times n\times p}$, and let $q = \min\{m, n\}$ and $r = \min\{\ell, n\}$. Denote the tensor singular values under the T–SVD (Definition 48) by $\varsigma_1(\cdot) \ge \cdots \ge \varsigma_q(\cdot) \ge 0$. Let the Fourier slices be*

$$\widehat{A}_k = \widehat{\mathcal{A}}(:,:,k), \qquad \widehat{B}_k = \widehat{\mathcal{B}}(:,:,k), \qquad k = 1, \dots, p.$$

Then, for each $i = 1, \dots, \min\{q, r\}$, the following bounds hold:

$$\varsigma_i(\mathcal{A}) \min_{1\le k\le p} \sigma_r(\widehat{B}_k) \;\le\; \varsigma_i(\mathcal{A} * \mathcal{B}) \;\le\; \varsigma_i(\mathcal{A}) \max_{1\le k\le p} \sigma_1(\widehat{B}_k). \tag{16}$$

Proof. Let $\mathcal{C} = \mathcal{A} * \mathcal{B}$. By the definition of the T–product and the DFT along the third mode (Definition 46), the Fourier slices satisfy

$$\widehat{\mathcal{C}}(:,:,k) = \widehat{A}_k \widehat{B}_k, \qquad k = 1, \dots, p.$$

For each fixed k, applying the matrix multiplicative singular-value bounds (Theorem 17) gives

$$\sigma_i(\widehat{A}_k)\,\sigma_r(\widehat{B}_k) \;\le\; \sigma_i(\widehat{A}_k\widehat{B}_k) \;\le\; \sigma_i(\widehat{A}_k)\,\sigma_1(\widehat{B}_k), \qquad i = 1, \dots, q.$$

By the definition of tensor singular values (Theorem 48),

$$\varsigma_i(\mathcal{A}) = \frac{1}{p}\sum_{k=1}^{p} \sigma_i(\widehat{A}_k), \qquad \varsigma_i(\mathcal{C}) = \frac{1}{p}\sum_{k=1}^{p} \sigma_i(\widehat{A}_k\widehat{B}_k).$$

Noting that for every k we have:

$$\sigma_1(\widehat{B}_k) \le \max_{1\le j\le p} \sigma_1(\widehat{B}_j),$$
$$\sigma_r(\widehat{B}_k) \ge \min_{1\le j\le p} \sigma_r(\widehat{B}_j),$$

Therefore, summing the inequalities over all slices, we obtain:

$$\varsigma_i(\mathcal{C}) \le \frac{1}{p}\sum_{k=1}^{p} \sigma_i(\widehat{A}_k)\,\sigma_1(\widehat{B}_k) \;\le\; \left(\max_{1\le k\le p} \sigma_1(\widehat{B}_k)\right) \frac{1}{p}\sum_{k=1}^{p} \sigma_i(\widehat{A}_k).$$

Similarly,

$$\varsigma_i(\mathcal{C}) \ge \frac{1}{p}\sum_{k=1}^{p} \sigma_i(\widehat{A}_k)\,\sigma_r(\widehat{B}_k) \;\ge\; \left(\min_{1\le k\le p} \sigma_r(\widehat{B}_k)\right) \frac{1}{p}\sum_{k=1}^{p} \sigma_i(\widehat{A}_k).$$

Combining both inequalities gives the result. □

In particular, if $\widehat{B}_k$ is rank-deficient for some k, then $\sigma_r(\widehat{B}_k) = 0$ and the lower bound in (16) becomes trivial. Future work could consider developing analogous principal angles that tighten these in practice but that is out of scope of our paper.

Tensor singular value bounds for T-products. We now combine the additive and multiplicative tensor singular value bounds to characterize the behaviour of tensors of the form

$$\mathcal{H} = \mathcal{A} * \mathcal{B} + \mathcal{C},$$

which we will later show captures the product structure existing in the third order derivatives of a neural network.

By the additive tensor singular value bound (Theorem 51), for any $i = 1, \dots, q$ we have

$$\varsigma_i(\mathcal{A} * \mathcal{B}) - \varsigma_1(\mathcal{C}) \;\le\; \varsigma_i(\mathcal{A} * \mathcal{B} + \mathcal{C}) \;\le\; \varsigma_i(\mathcal{A} * \mathcal{B}) + \varsigma_1(\mathcal{C}).$$

Applying the multiplicative tensor singular value bounds (Theorem 52) to the T–product term $\mathcal{A} * \mathcal{B}$ yields

$$\varsigma_i(\mathcal{A}) \min_{1\le k\le p} \sigma_r(\widehat{B}_k) \;\le\; \varsigma_i(\mathcal{A} * \mathcal{B}) \;\le\; \varsigma_i(\mathcal{A}) \max_{1\le k\le p} \sigma_1(\widehat{B}_k).$$

Combining the two inequalities gives the composite bounds for all $i = 1, \dots, \min\{q, r\}$:

$$\varsigma_i(\mathcal{A} * \mathcal{B} + \mathcal{C}) \;\le\; \varsigma_i(\mathcal{A}) \max_{1\le k\le p} \sigma_1(\widehat{B}_k) \;+\; \varsigma_1(\mathcal{C}), \tag{17}$$

and

$$\varsigma_i(\mathcal{A} * \mathcal{B} + \mathcal{C}) \;\ge\; \varsigma_i(\mathcal{A}) \min_{1\le k\le p} \sigma_r(\widehat{B}_k) \;-\; \varsigma_1(\mathcal{C}). \tag{18}$$

As in the purely multiplicative case, the lower bound (18) may become trivial if one or more Fourier slices $\widehat{B}_k$ are rank-deficient, in which case $\sigma_r(\widehat{B}_k) = 0$. Nevertheless, the upper bound (17) always holds and shows that the leading tensor singular value of $\mathcal{A} * \mathcal{B} + \mathcal{C}$ is controlled by the spectrum of $\mathcal{A}$ and $\mathcal{B}$, up to an additive contribution from $\mathcal{C}$.

These bounds provide the key mechanism by which spectral properties of individual weight matrices can be used to control the tensor singular values of third-order derivative blocks. We now illustrate this structure explicitly for MLP.

J.3.7. Third Order Analysis: Application to Neural Networks

We now present an example analysis of a four-layer MLP to illustrate how modifying the spectral values of the weights of a network related to L_2 and thereby the convergence rate of second-order methods.

Example 53 (Multi-layer Perceptron(MLP)). Consider the four-layer MLP:

$$f(x) = \theta_4^\top \, \phi_3\Big(\theta_3 \, \phi_2\big(\theta_2 \, \phi_1(\theta_1 x)\big)\Big),$$

where the weight matrices have dimensions

$$\theta_1 \in \mathbb{R}^{m_1 \times d}, \qquad \theta_2 \in \mathbb{R}^{m_2 \times m_1}, \qquad \theta_3 \in \mathbb{R}^{m_3 \times m_2}, \qquad \theta_4 \in \mathbb{R}^{m_3}.$$

Define the layer pre-activations and activations as

$$\begin{aligned} h_1 &= \theta_1 x, & z_1 &= \phi(h_1), \\ h_2 &= \theta_2 z_1, & z_2 &= \phi(h_2), \\ h_3 &= \theta_3 z_2, & z_3 &= \phi(h_3), \end{aligned}$$

so that the network output simplifies to

$$f(x) = \theta_4^\top z_3.$$

Let the diagonal Jacobian matrices of the nonlinearity be

$$\begin{aligned} D_{z_1} &= \mathrm{Diag}(\phi'(h_1)) \in \mathbb{R}^{m_1 \times m_1}, & D_{z_2} &= \mathrm{Diag}(\phi'(h_2)) \in \mathbb{R}^{m_2 \times m_2}, \\ D_{z_3} &= \mathrm{Diag}(\phi'(h_3)) \in \mathbb{R}^{m_3 \times m_3}, & D'_{z_3} &= \mathrm{Diag}(\phi''(h_3)) \in \mathbb{R}^{m_3 \times m_3}. \end{aligned}$$

We consider the mixed third derivative

$$\nabla^3_{\theta_3,\,\theta_1,\,\theta_4} f(x) \in \mathbb{R}^{m_3 \times (m_1 d) \times (m_3 m_2)},$$

whose frontal slices correspond to derivatives with respect to each entry $(\theta_3)_{ab}$.

To begin, differentiate f with respect to θ_4. Since $f = \theta_4^\top z_3$, it follows that:

$$\frac{\partial f}{\partial \theta_4} = z_3.$$

Next, we differentiate f with respect to θ_1 and θ_4. Applying the chain rule,

$$\frac{\partial z_3}{\partial z_2} = D_{z_3}\theta_3, \qquad \frac{\partial z_2}{\partial z_1} = D_{z_2}\theta_2, \qquad \frac{\partial z_1}{\partial \theta_1} = D_{z_1}(x^\top \otimes I_{m_1}),$$

and therefore:

$$\frac{\partial^2 f}{\partial \theta_1\, \partial \theta_4} = D_{z_3}\theta_3 D_{z_2}\theta_2 D_{z_1}\,(x^\top \otimes I_{m_1}),$$

$$\theta_3 = \begin{bmatrix} (\theta_3)_{11} & (\theta_3)_{12} & \cdots & (\theta_3)_{1m_2} \\ (\theta_3)_{21} & (\theta_3)_{22} & \cdots & (\theta_3)_{2m_2} \\ \vdots & \vdots & \ddots & \vdots \\ (\theta_3)_{m_3 1} & (\theta_3)_{m_3 2} & \cdots & (\theta_3)_{m_3 m_2} \end{bmatrix}, \qquad \theta_3 \in \mathbb{R}^{m_3 \times m_2}.$$

Finally, we differentiate f with respect to $(\theta_3, \theta_1, \theta_4)$. For each entry (a, b) define the basis matrix

$$E_{ab} \in \mathbb{R}^{m_3 \times m_2}, \qquad [E_{ab}]_{ij} = \begin{cases} 1, & i = a,\ j = b, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, differentiating gives:

$$\frac{\partial \theta_3}{\partial (\theta_3)_{ab}} = E_{ab} \in \mathbb{R}^{m_3 \times m_2}, \qquad \frac{\partial D_{z_3}}{\partial (\theta_3)_{ab}} = \mathrm{Diag}\big(D'_{z_3}(E_{ab} z_2)\big) \ \in \ \mathbb{R}^{m_3 \times m_3}$$

where $\mathrm{Diag}(\cdot)$ denotes the diagonal matrix formed from a vector.

Consequently, each frontal slice of the third derivative tensor admits the decomposition

$$\frac{\partial^3 f}{\partial (\theta_3)_{ab}\, \partial \theta_1\, \partial \theta_4} = \mathcal{C}(:,:,k) \ + \ \mathcal{A}(:,:,k)\, \mathcal{B}(:,:,k), \tag{19}$$

where

$$\mathcal{A}(:,:,k) = D_{z_3} E_{ab} D_{z_2}, \qquad \mathcal{B}(:,:,k) = \theta_2 D_{z_1}(x^\top \otimes I_{m_1}),$$

and

$$\mathcal{C}(:,:,k) = \mathrm{Diag}\big(D'_{z_3}(E_{ab} z_2)\big)\, \theta_3 D_{z_2}\theta_2 D_{z_1}(x^\top \otimes I_{m_1}).$$

Here $k \leftrightarrow (a, b)$ indexes the entries of θ_3, and $p = m_3 m_2$ is the total number of such slices.

Stacking the matrices over all (a, b) along the third dimension yields the third-order tensor

$$\mathcal{H} = \nabla^3_{\theta_3,\theta_1,\theta_4} f(x) \in \mathbb{R}^{m_3 \times (m_1 d) \times p}.$$

We emphasize that $\mathcal{H}$ represents the coordinate-slice (basis-dependent) representation of the third-order derivative tensor with respect to the standard basis $\{E_{ab}\}$ of θ_3.

Let $w := \mathrm{vec}(\theta_3) \in \mathbb{R}^p$ with $p = m_3 m_2$. Since $\nabla^3_{\theta_3,\theta_1,\theta_4} f(x)$ is trilinear in θ_3, it admits a directional representation.

Given a set of unit vectors $\{\phi_k\}_{k=1}^p \subset \mathbb{C}^p$, we define the directionally parameterized third-order tensor $\mathcal{T}(x) \in \mathbb{R}^{m_3 \times (m_1 d) \times p}$ by

$$\mathcal{T}(:,:,k) := \nabla^3_{w,\theta_1,\theta_4} f(x)[\phi_k] = \sum_{j=1}^{p} \mathcal{H}(:,:,j)\, (\phi_k)_j.$$

We emphasize that this construction does not change the order of differentiation: $\mathcal{T}$ represents a change of basis of the third-order derivative tensor along the θ_3 mode, rather than a different differential object.

Throughout this example, we take $\{\phi_k\}$ to be the unitary Fourier basis associated with the normalized DFT matrix F_p. Under this choice, $\mathcal{T}$ is exactly the unitary DFT of $\mathcal{H}$ along the third mode.

Accordingly, the tensors $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are interpreted as the directional lifts of their coordinate-slice counterparts under the same unitary basis, so that their Fourier slices correspond to the associated linear combinations along the third mode.

In the Fourier-domain representation associated with the third mode, we have

$$\widehat{\mathcal{T}}(:,:,k) = \widehat{\mathcal{A}}(:,:,k)\,\widehat{\mathcal{B}}(:,:,k) + \widehat{\mathcal{C}}(:,:,k), \qquad k = 1,\dots,p.$$

By the definition of the T–product, this identity is equivalent to the exact tensor decomposition

$$\mathcal{T} = \mathcal{A} * \mathcal{B} + \mathcal{C}.$$

As a consequence, the tensor singular values of the directionally parameterized tensor $\mathcal{T}$ can be bounded by combining the multiplicative tensor singular value bounds for $\mathcal{A} * \mathcal{B}$ (Theorem 52) with the additive tensor singular value bounds for the perturbation $\mathcal{C}$ (Theorem 51).

Since $\mathcal{T}$ is obtained from $\mathcal{H}$ via a unitary transform along the third mode, their tensor singular values coincide. Therefore, the same bound applies to the original third-order derivative tensor $\mathcal{H}$.

Table 29 reports a numerical verification of the T–SVD bounds derived in the MLP example (Example 53). To test the effect of spectral scaling, we introduce a scalar factor $\alpha > 0$ and construct a scaled tensor

$$\mathcal{H}(\alpha) \;:=\; (\alpha\,\mathcal{A}) * \mathcal{B} + \mathcal{C},$$

where the scaling is applied exclusively to $\mathcal{A}$, while $\mathcal{B}$ and $\mathcal{C}$ are held fixed. For each value of α, the tensor $\mathcal{H}(\alpha)$ is explicitly reconstructed via the T–product.

For each $\mathcal{H}(\alpha)$, we compute the maximum tensor singular value $\varsigma_1(\mathcal{H}(\alpha))$ using the T–SVD definition, together with the corresponding lower and upper bounds

$$\mathrm{LB}_1(\alpha) = \varsigma_1(\mathcal{A})\,\min_k \sigma_r(\widehat{B}_k) - \varsigma_1(\mathcal{C}), \qquad \mathrm{UB}_1(\alpha) = \varsigma_1(\mathcal{A})\,\max_k \sigma_1(\widehat{B}_k) + \varsigma_1(\mathcal{C}),$$

as given by (18) and (17).

From this table we observe that, as the scaling factor α increases, both the lower bound $\mathrm{LB}_1(\alpha)$ and the upper bound $\mathrm{UB}_1(\alpha)$ increase correspondingly. This behavior is consistent with the composite tensor singular value bounds in (17)–(18), which predict that scaling the multiplicative component $\mathcal{A}$ proportionally enlarges the admissible range of $\varsigma_1(\mathcal{H}(\alpha))$ when $\mathcal{B}$ and $\mathcal{C}$ are held fixed.

Table 29. Verification of lower and upper bounds for the top tensor singular value $\varsigma_1(\mathcal{H}(\alpha))$ under consistent scaling of $\mathcal{A}$.

α	$\varsigma_1(\mathcal{H}(\alpha))$	$\mathrm{LB}_1(\alpha)$	$\mathrm{UB}_1(\alpha)$
1	1.27	−0.20	2.15
10	13.34	0.18	19.27
100	134.15	3.94	190.52
1000	1342.27	41.62	1903.03
10000	13423.46	418.43	19028.13
100000	134235.38	4186.45	190279.08

Based on this example we can provide a number of remarks on when control of the spectral values of the weights alone are useful for controlling L_2.

Remark 54 (Conditions for Effective L_2 Control via Spectral Inflation). The preceding analysis establishes that the Hessian Lipschitz constant L_2 upper bounds the largest tensor singular value of $\nabla^3 f$ (Theorem 49), and that the tensor singular values of composite expressions $\mathcal{A} * \mathcal{B} + \mathcal{C}$ satisfy the bounds (17)–(18). We now discuss the conditions under which inflating weight singular values translates into effective control of L_2.

When spectral inflation is effective The upper bound (17) shows that $\varsigma_i(\mathcal{A} * \mathcal{B} + \mathcal{C}) \leq \varsigma_i(\mathcal{A}) \max_k \sigma_1(\widehat{B}_k) + \varsigma_1(\mathcal{C})$. Inflating the singular values of weight matrices appearing in $\mathcal{A}$ and $\mathcal{B}$ directly increases the right-hand side, thereby *permitting* a larger L_2. This upper bound argument is tight when:

(i) The activation functions have bounded second derivatives (e.g., $\|\phi''\|_\infty < \infty$), ensuring that the additive term $\mathcal{C}$ containing D'_{z_ℓ} remains controlled;
(ii) The network operates in a regime where the multiplicative structure $\mathcal{A} * \mathcal{B}$ dominates the third-order derivative tensor.

Limitations of the lower bound The lower bound (18) is subtractive: $\varsigma_i(\mathcal{A} * \mathcal{B} + \mathcal{C}) \geq \varsigma_i(\mathcal{A}) \min_k \sigma_r(\widehat{B}_k) - \varsigma_1(\mathcal{C})$. This bound becomes vacuous when either (i) some Fourier slice $\widehat{B}_k$ is rank-deficient, or (ii) the perturbation term $\varsigma_1(\mathcal{C})$ dominates. In such cases, inflating weight spectra does not guarantee a proportional increase in the tensor singular values of $\nabla^3 f$, and hence does not guarantee increased L_2.

Sufficient conditions for guaranteed L_2 increase. A sufficient condition for spectral inflation to increase L_2 is that the perturbation term satisfies

$$\varsigma_1(\mathcal{C}) \leq \varsigma_1(\mathcal{A}) \min_k \sigma_r(\widehat{B}_k) - \delta \tag{20}$$

for some margin $\delta > 0$, and all Fourier slices $\widehat{B}_k$ have full column rank. Under this condition, increasing $\varsigma_1(\mathcal{A})$ by a factor $\alpha > 1$ yields

$$\varsigma_1(\mathcal{A} * \mathcal{B} + \mathcal{C}) \geq \alpha \cdot \varsigma_1(\mathcal{A}) \min_k \sigma_r(\widehat{B}_k) - \varsigma_1(\mathcal{C}) \geq (\alpha - 1)\delta + \varsigma_1(\mathcal{A} * \mathcal{B} + \mathcal{C})\big|_{\alpha=1},$$

demonstrating a strict increase in the tensor singular value and hence in L_2.

Practical implications In practice, condition (20) is most likely satisfied in the following regimes:

- *Smooth activations*: When $\phi'' \approx 0$ (e.g., near-linear regions of tanh or GELU), the term $\mathcal{C}$ containing D'_{z_ℓ} is small.
- *Moderate depth*: In shallow networks, fewer compositional terms appear in $\mathcal{C}$, reducing its magnitude relative to $\mathcal{A} * \mathcal{B}$.
- *Well-conditioned intermediate representations*: When $\min_k \sigma_r(\widehat{B}_k)$ is bounded away from zero, the multiplicative term provides a strong baseline.

We acknowledge that our theoretical framework provides *permissive* rather than *prescriptive* guarantees: inflating weight spectra *allows* L_2 to be larger but does not unconditionally *force* it to be larger. The empirical effectiveness of SpecDef observed in our experiments suggests that practical networks often operate in regimes where condition (20) approximately holds, though a complete characterization of these regimes remains an important direction for future theoretical work.