INFLUENCE FUNCTIONS FOR NON-STATIONARY TRAINING CHECKPOINTS VIA THE TRUE HESSIAN

Anonymous authors
Paper under double-blind review

ABSTRACT

We generalize influence functions to non-stationary training checkpoints by constructing a local quadratic model of the training objective under a trust-region proximity penalty, which recovers the classical estimator and, in form, the Proximal Bregman Response Function (PBRF). Unlike PBRF, our estimator keeps the true Hessian, including its negative curvature, and requires solving two linear systems in the indefinite Hessian damped by a trust-region parameter λ. We give a selection rule for λ that keeps the damped Hessian invertible with a safety margin and the quadratic model within an estimated trust radius, while controlling how much curvature the damping discards. We solve both systems with MINRES, and give a Lanczos–Taylor approximation that reuses the Krylov basis already computed for λ-selection. We evaluate on four-layer MLPs trained on UCI Concrete, MNIST, and FashionMNIST. On UCI Concrete, our estimator attains the highest Linear Datamodeling Score (LDS) for a 10-model ensemble and matches the strongest baseline, EKFAC-IF, for a single model, ahead of PBRF and ASTRA-IF. On MNIST it is second to PBRF and ahead of ASTRA-IF, EKFAC-IF, TracIn, and TRAK. On FashionMNIST it ties PBRF and ASTRA-IF and is ahead of EKFAC-IF, TracIn, and TRAK.

1 INTRODUCTION

Training data attribution (TDA) identifies which training examples are responsible for a given model's prediction. TDA has been used to explain model predictions, relabel mislabeled training examples, audit fairness, and design data augmentation strategies, among other applications (Koh & Liang, 2017; Han et al., 2020; Kong et al., 2022; Brunet et al., 2019; Schulam & Saria, 2019; Lee et al., 2020). When a language model produces a hallucinated output, attribution can distinguish a single culpable training example from a broader pattern spread across the corpus. TDA can also indicate that the training data is not the main reason for the error, in which case the cause may instead be attributed to other factors that include the model's architecture, its decoding procedure, or the training algorithm itself (Bengio et al., 2015; Holtzman et al., 2020).

Influence functions estimate the effect of upweighting or downweighting a training example on a model's prediction, without the cost of retraining. Koh & Liang (2017) introduced this technique from statistics to deep learning, approximating this effect for a training example z_m by computing

$$\mathcal{I}_f(z_m) = -H_{\theta^\star}^{-1} \nabla_\theta L(z_m, \theta^\star), \tag{1}$$

where $\theta^\star$ are the parameters that minimize the training objective $\mathcal{J}$, L is the per-example loss, and $H_{\theta^\star} = \nabla^2_\theta \mathcal{J}(\theta^\star, \mathcal{D})$ is the Hessian of $\mathcal{J}$ at $\theta^\star$. The derivation of Equation 1 requires $\theta^\star$ to be an exact stationary point of $\mathcal{J}$.

Neural networks' training is often terminated before reaching a stationary point of the training objective $\mathcal{J}$, to avoid overfitting or due to limited computational resources (Bengio, 2012). The Proximal Bregman Response Function (PBRF) (Bae et al., 2022) replaces $\mathcal{J}$ with a Bregman divergence over the model's outputs, for which the parameters θ_s, at which training was terminated, exactly minimize this new objective. Solving for the PBRF yields a linear system involving the Gauss-Newton Hessian G, which is positive semi-definite, approximating the nonnegative portion of H's spectrum (Bae et al., 2022; Martens, 2020).

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

Negative directions in the loss landscape can help speed up training by guiding it away from a saddle point, a stationary point of the training objective from which the loss can still be decreased along at least one direction (Dauphin et al., 2014). Failing to recognize a saddle point can be mistaken for reaching a genuine minimum, and the surrounding plateau can slow training considerably (Dauphin et al., 2014). The Hessians of deep networks are indefinite in practice, possessing a mix of positive, negative, and near-zero eigenvalues (Sagun et al., 2017). H's spectrum can capture these negative directions, whereas the Gauss-Newton Hessian G is constrained to be positive semi-definite.

Solving for $\mathcal{I}_\lambda(z_m)$ at the scale of a modern neural network can be done using Krylov subspace methods. Since H is symmetric and indefinite, MINRES is the natural choice, building a solution within a growing subspace at a cost per iteration of a single matrix-vector product with H (Paige & Saunders, 1975). A dense cluster of eigenvalues near zero, characteristic of the Hessians of deep networks (Sagun et al., 2017; Papyan, 2019), slows MINRES's convergence to the solution and can cause its basis vectors to lose orthogonality, corrupting the solution it computes (Paige, 1976). Preconditioning can address both of these effects by improving how well-separated the matrix's eigenvalues appear to the solver.

Our contributions are as follows:

1. **A generalized influence function** (Section 3.1) constructed from a local quadratic model of the training objective under a trust-region proximity penalty λ, yielding two linear systems involving $H + \lambda I$.
2. **A λ-selection algorithm** (Section 3.3) that keeps $H + \lambda I$ invertible while balancing the preservation of H's curvature against the validity of this quadratic approximation.
3. **A solver** (Section 3.4) based on MINRES, with a Lanczos–Taylor approximation that reuses the Krylov basis from λ-selection and a Chebyshev polynomial preconditioner for λ close to the nearest pole. A residual-fraction test, computed before any scores, chooses between the approximation and a direct solve. On UCI Concrete, the approximation improves LDS over the direct solve at the same λ.
4. **Empirical evaluation** (Section 4) against PBRF, EKFAC-IF, ASTRA-IF, TracIn, and TRAK on UCI Concrete, MNIST, and FashionMNIST, using the Linear Datamodeling Score (LDS). Our estimator attains the highest LDS on UCI Concrete for a 10-model ensemble and is comparable to the best baseline for a single model. On MNIST it is second to PBRF, and on FashionMNIST it ties PBRF and ASTRA-IF.

2 Related Work

Training Data Attribution Methods. Identifying which training examples are responsible for a given model prediction has been studied through several empirical approaches. TracIn (Pruthi et al., 2020) estimates a training example's influence on a query prediction by accumulating the alignment between the training and query loss gradients across checkpoints saved during training. TRAK (Park et al., 2023) linearizes the model around its trained parameters and estimates influence via a projected-gradient regression, averaged across an ensemble of independently-trained models. Other approaches infer attribution from retraining outcomes. Datamodels (Ilyas et al., 2022) fit a linear regression predicting a model's output on a query example directly from which training examples were included in its training set, using outcomes collected from many independently-trained models. Data Shapley (Ghorbani & Zou, 2019) instead values each training example by its average marginal contribution to model performance across many randomly sampled subsets, drawing on the Shapley value from cooperative game theory.

Influence Functions and Non-Convergence. Koh & Liang (2017) introduced influence functions to deep learning, approximating the effect of removing a training example as the solution of a linear system that requires the model's parameters to be an exact stationary point of the training objective. Modern training routinely stops short of convergence, whether by early stopping or a single pass over a large corpus (Muennighoff et al., 2023). Bae et al. (2022) decompose the error between influence functions and true leave-one-out retraining into five components, one of which arises specifically from this non-convergence. Bae et al. (2024) address this non-convergence component by unrolling differentiation through the actual training trajectory.

Solving for Influence via Iterative Methods. Computing an influence score requires solving a linear system involving either the true Hessian or the Gauss-Newton Hessian. Neumann-series methods approximate the inverse of the Hessian directly through a truncated power series. LiSSA (Agarwal et al., 2017), used by Koh & Liang (2017) for the true Hessian, gives a stochastic, recursive approximation of the inverse-Hessian-vector product (iHVP) requiring single-example Hessian-vector products per iteration.

Krylov subspace methods build a solution within a growing subspace spanned by repeated matrix-vector products. Conjugate gradients (CG) (Hestenes & Stiefel, 1952) requires a symmetric positive-definite matrix. Koh & Liang (2017) add a damping term to the indefinite true Hessian to achieve this. Bae et al. (2022) apply the same solver to the Gauss-Newton Hessian, which is positive semi-definite, requiring damping to ensure strict invertibility. MINRES (Paige & Saunders, 1975) extends CG's applicability to symmetric matrices, without requiring positive-definiteness, and guarantees monotonic residual decrease. For positive-definite matrices, MINRES's convergence rate takes the same form as CG's, depending on the matrix's condition number. For indefinite matrices, convergence instead depends on how well-separated every eigenvalue is from zero relative to the matrix's overall eigenvalue range, converging slowly when some eigenvalues lie close to zero, motivating the use of a preconditioner to improve this eigenvalue separation directly.

Scaling Influence Functions via Hessian Approximation. A separate line of work makes the linear system satisfied by the influence function tractable by approximating the curvature matrix itself. Grosse et al. (2023) scale influence functions to language models with up to 52 billion parameters using the Eigenvalue-corrected Kronecker-Factored Approximate Curvature method (EK-FAC) (George et al., 2018). EK-FAC approximates the Gauss-Newton Hessian with a block-diagonal Kronecker factorization, assuming independence between layers and enabling eigendecomposition one layer at a time. ASTRA (Wang et al., 2025) applies the LiSSA iteration to the damped Gauss-Newton Hessian, preconditioning it with EK-FAC to obtain a more accurate inverse-Hessian-vector product.

3 Influence Functions Beyond Stationarity: From Derivation to Scalable Computation

3.1 Influence Functions at Non-Stationary Checkpoints

Let $\theta \in \mathbb{R}^d$ denote model parameters, z a data point, and $L(z, \theta)$ a per-example loss, assumed twice continuously differentiable in θ for every z. We do not assume $L(z, \cdot)$ is convex, as empirical studies of trained neural networks consistently find genuinely negative Hessian eigenvalues (Sagun et al., 2017; Ghorbani et al., 2019). The training objective $\mathcal{J}(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(z_i, \theta)$ is twice continuously differentiable.

The classical influence function (Koh & Liang, 2017) requires θ to be an exact stationary point of $\mathcal{J}$. We consider instead a checkpoint θ_s, the parameters produced by the optimization trajectory at some intermediate step s, with $g_s \equiv \nabla_\theta \mathcal{J}(\theta_s)$ generally nonzero, at which the classical formulation does not apply.

We construct, in place of $\mathcal{J}$ itself, a local model $\Phi_{\varepsilon,\lambda}$ of $\mathcal{J}$ around θ_s that admits an exact stationarity condition, and apply the Implicit Function Theorem (IFT) to $\Phi_{\varepsilon,\lambda}$ instead. A second-order Taylor expansion of $\mathcal{J}$ around θ_s is not sufficient on its own. Its unconstrained stationary point can lie arbitrarily far from θ_s whenever $g_s \neq 0$, extrapolating the local approximation into a region it was never fit to describe. We add an explicit proximity penalty, the Lagrangian relaxation of a trust-region constraint (Conn et al., 2000) restricting θ to a neighborhood of θ_s. Let $z_m \in \mathcal{D}$ be a training example whose influence we wish to estimate. We retain the query loss $L(z_m, \cdot)$ exactly rather than expanding it, since, as we show in §3.2, the stationarity condition used to define our estimator does not require it to be approximated.

The resulting local model is given by,

$$\Phi_{\varepsilon,\lambda}(\theta) = \mathcal{J}(\theta_s) + g_s^\top(\theta - \theta_s) + \frac{1}{2}(\theta - \theta_s)^\top H(\theta - \theta_s) + \varepsilon L(z_m, \theta) + \frac{\lambda}{2}\|\theta - \theta_s\|^2, \quad (2)$$

where $H \equiv \nabla^2_\theta \mathcal{J}(\theta_s)$ is the true Hessian, not assumed positive semi-definite, and $\lambda \geq 0$ is the Lagrange multiplier associated with the trust-region constraint. In §3.2, we apply the IFT to $\Phi_{\varepsilon,\lambda}$'s stationarity condition, yielding two linear systems in $(H + \lambda I)$ whose solutions define our estimator. §3.3 chooses λ so that $(H+\lambda I)$ is numerically well-posed while preserving the curvature information carried by H.

3.2 Computing the Influence Function via Two Linear Systems

We seek to compute the sensitivity of the stationary point $\widehat{\theta}_\lambda(\varepsilon)$ of $\Phi_{\varepsilon,\lambda}$ to ε, given by $\mathcal{I}_\lambda(z_m) \equiv d\widehat{\theta}_\lambda(\varepsilon)/d\varepsilon\Big|_{\varepsilon=0}$. The point $\widehat{\theta}_\lambda(\varepsilon)$ satisfies

$$F(\theta, \varepsilon) \equiv \nabla_\theta \Phi_{\varepsilon,\lambda}(\theta) = g_s + (H + \lambda I)(\theta - \theta_s) + \varepsilon \nabla_\theta L(z_m, \theta) = 0. \tag{3}$$

The IFT computes $\mathcal{I}_\lambda(z_m)$ by expanding equation 3 around a root $(\theta, 0)$ of equation 3. We first establish that such a root exists.

Let y_λ be the displacement of the solution θ from θ_s, satisfying $\theta = \theta_s - y_\lambda$. Setting $\varepsilon = 0$ in equation 3 then gives

$$(H + \lambda I)\, y_\lambda = g_s. \tag{4}$$

Applying the IFT to equation 3 at $(\theta_s - y_\lambda, 0)$ yields $\widehat{\theta}_\lambda(\varepsilon)$ for ε near 0. Differentiating equation 3 with respect to ε at $\varepsilon = 0$ gives a second linear system,

$$(H + \lambda I)\, \mathcal{I}_\lambda(z_m) = -\nabla_\theta L(z_m,\, \theta_s - y_\lambda). \tag{5}$$

Systems equation 4 and equation 5 share the coefficient matrix $H + \lambda I$. In §3.3, we choose λ so that $H + \lambda I$ is invertible, guaranteeing both a unique solution to equation 4 and the validity of the IFT step above. In §3.4, we exploit this shared coefficient matrix, together with its indefinite structure, to solve both systems efficiently. We verified the derivation numerically against a direct Newton solve of the exact nonlinear stationarity condition: the discrepancy shrinks linearly in ε, as the IFT predicts.

Special cases. Our estimator recovers two existing approaches:

- Retaining H and evaluating at a genuine stationary point $\theta_s = \theta^\star$, at which $g_s = 0$ holds exactly, recovers the influence function $-H^{-1}\nabla_\theta L(z_m, \theta^\star)$ of Koh & Liang (2017) as $\lambda \to 0^+$; their own implementation adds a damping term of the same form as λI for non-convex models.
- Setting $H = G$, the Gauss-Newton approximation to the Hessian, and $g_s = 0$ recovers the closed-form Proximal Bregman Response Function (PBRF) estimator $-(G + \lambda I)^{-1}\nabla_\theta L(z_m, \theta_s)$ of Bae et al. (2022), obtained by replacing the training objective with a Bregman divergence over model outputs, for which θ_s is exactly stationary by construction.

Replacing H with G means discarding any negative curvature H may carry, since G is positive semi-definite by construction. Negative curvature directions can dominate the loss landscapes of high-dimensional non-convex models, corresponding to saddle points rather than local minima (Dauphin et al., 2014).

Decomposition. The influence estimate decomposes as

$$\mathcal{I}_\lambda(z_m) = -x_\lambda(z_m) + e_\lambda(z_m), \text{ where,} \tag{6}$$

$$x_\lambda(z_m) \equiv (H+\lambda I)^{-1}\nabla_\theta L(z_m, \theta_s), \text{ and } e_\lambda(z_m) \equiv (H+\lambda I)^{-1}\Big[\nabla_\theta L(z_m, \theta_s) - \nabla_\theta L(z_m, \theta_s - y_\lambda)\Big]. \tag{7}$$

$x_\lambda(z_m)$ is the influence estimate obtained by evaluating the query gradient at θ_s directly, with no adjustment for $g_s \neq 0$. $e_\lambda(z_m)$ is the correction due to θ_s not being stationary, given by the difference between the query gradient evaluated at θ_s and at the shifted point $\theta_s - y_\lambda$. It is an exact finite difference of gradients rather than a linearized Hessian-vector product, and is precisely the correction for the non-convergence effect that Bae et al. (2022) identify as a source of error in their own decomposition of estimation error.

3.3 Choosing λ to Balance Curvature Preservation and Trust-Region Validity

We derive a stability interval $[\lambda_{\min}, \lambda_{\max}]$ and choose λ within it so that the quadratic approximation of $\mathcal{J}$ is accurate while preserving the real curvature information H carries. Writing H's spectral decomposition as $H = \sum_i \theta_i v_i v_i^\top$, the solution to System 4 is $y_\lambda = \sum_i c_i v_i$, where $c_i \equiv (g_s \cdot v_i)/(\theta_i + \lambda)$ denotes direction i's coefficient. A pole at $\lambda = -\theta_i$ makes c_i unbounded regardless of direction i's informativeness, while large λ shrinks every c_i toward zero, discarding curvature.

Upper bound $\lambda_{\max}$. Protecting the smallest eigenvalue magnitude we wish to preserve, θ_{ref}, with a tolerance $\varepsilon_{\text{damp}}$ for the largest fraction of θ_{ref}'s coefficient we allow to be lost, gives $\lambda_{\max} \equiv \theta_{\text{ref}} \cdot \frac{\varepsilon_{\text{damp}}}{1-\varepsilon_{\text{damp}}}$. We place θ_{ref} at the boundary between the bulk and the outliers of H's spectrum, a structure documented for trained networks (Sagun et al., 2017; Papyan, 2019), estimated via stochastic Lanczos quadrature (Appendix B).

Lower bound $\lambda_{\min}$. λ must avoid two distinct failure modes. A pole at $\lambda = -\theta_i$ makes c_i unbounded. A cluster of H's eigenvectors with negative eigenvalues near $-\lambda$ spans a subspace along which $(H + \lambda I)^{-1}$ scales any vector's component by a factor of exactly $1/(\theta_i + \lambda)$, large whenever $\theta_i + \lambda$ is small. Since $H - G$ is precisely the additional information H contributes over G (§3.2), disproportionately amplifying $H - G$'s own small component in this subspace can move the resulting estimate away from the correct answer rather than toward it. We guard against both failure modes with a single margin, $\lambda_{\text{cluster}} \equiv \tau_{\text{safe}} + |\theta_{\min}|$, where $\theta_{\min}$ is H's most negative eigenvalue and τ_{safe} controls how close $\theta_{\min} + \lambda$ may come to zero (Appendix B).

The quadratic model (Equation 2) is accurate only within a trust radius $\Delta_{\max}$, derived from a Lipschitz bound on $\mathcal{J}$'s third derivative near θ_s (Conn et al., 2000) (Appendix B). The trust-region requirement is then $\lambda \geq \lambda_{\text{trust}}$, the smallest λ satisfying $\|y_\lambda\| \leq \Delta_{\max}$. In practice g_s is typically estimated rather than known exactly, for instance from a minibatch (Bottou et al., 2018), giving a noise floor λ_{floor} (Appendix B). $\lambda_{\min}$ is the largest of λ_{floor}, λ_{cluster}, and λ_{trust}.

Algorithm. We find λ_{trust} by solving the secular equation for a trust-region step, $\|y_\lambda\|^2 - \Delta_{\max}^2 = 0$, via a single Lanczos process started from g_s (Gould et al., 1999; Moré & Sorensen, 1983). This gives a closed-form proxy for $\|y_\lambda\|^2$ that can be evaluated at any λ without further Hessian-vector products. The proxy is monotonic past λ_{cluster}, so its root is found by bisection. Requiring $\lambda \geq \lambda_{\text{cluster}}$ can push $\lambda_{\min}$ above the curvature-preservation ceiling $\lambda_{\max}$. Rather than clip to $\lambda_{\max}$ and proceed at an unsafe λ, we relax $\varepsilon_{\text{damp}}$ exactly as far as needed for $\lambda_{\max}$ to reach $\lambda_{\min}$, with no fixed relaxation ceiling. Since $\|y_\lambda\|$ decreases toward zero as λ grows, λ_{trust} is always finite, so a λ satisfying every requirement always exists. We report the relaxed $\varepsilon_{\text{damp}}$ as a per-checkpoint diagnostic of how much curvature had to be given up.

Evaluation point. The shift in System 5 is taken along the direction of the System 4 solution, rescaled to the trust radius: $\tilde{y}_\lambda = \Delta_{\max} y_\lambda / \|y_\lambda\|$. When λ is set by λ_{cluster} rather than λ_{trust}, $\|y_\lambda\| < \Delta_{\max}$, and this places the evaluation point at the largest displacement for which the quadratic model is trusted. We use $\tilde{y}_\lambda$ in place of y_λ in Equation 5 unless stated otherwise.

3.4 Solving the Systems: Efficient and Stable Computation

We solve Systems 4 and 5 with MINRES (Paige & Saunders, 1975), since $H + \lambda I$ is symmetric and not guaranteed positive definite. Each MINRES iteration costs one Hessian-vector product. The choice of λ in §3.3 keeps every eigenvalue of $H + \lambda I$ at least τ_{safe} away from zero, so the system is well conditioned and plain MINRES converges in few iterations at the selected λ: about 13 Hessian-vector products per solve to a relative residual of 10^{-5} on UCI Concrete (Appendix A.6).

Chebyshev preconditioning near the pole. When λ must lie close to $|\theta_{\min}|$, the dense bulk of $H + \lambda I$'s spectrum near zero (Sagun et al., 2017; Papyan, 2019) slows MINRES (Driscoll & Braun, 2022). For this regime we provide a preconditioner. Using $P^{-1} = |H + \lambda I|^{-1}$ would make preconditioned MINRES converge in at most two steps (Vecharynski & Knyazev, 2013). We approximate it by a degree-q polynomial $p(H + \lambda I)$ with $p(\nu) \approx 1/\max(|\nu|, \lambda_{\text{cap}})$, fitted at Chebyshev nodes on the extreme eigenvalues of $H + \lambda I$ (Trefethen, 2019), with a cap $\lambda_{\text{cap}} = 10\lambda$

that removes the singularity at zero from the fitting target (Appendix C). The cap changes only the preconditioner, never the system being solved, so the preconditioner affects cost but not the solution. Each application costs q Hessian-vector products ($q = 80$), so it pays off only when it reduces the iteration count by more than a factor of $q + 1$. At the selected λ on UCI Concrete it does not (Appendix A.6), and we use plain MINRES there.

One solve per query. Attribution scores are reported against a measurement f_q on query q: $\text{score}(z_i; q) = \nabla_\theta f_q(\theta_s)^\top \mathcal{I}_\lambda(z_i)$. Solving System 5 once per training example would cost N solves. Since $(H + \lambda I)^{-1}$ is symmetric,

$$\text{score}(z_i; q) = -w_q^\top \nabla_\theta L(z_i, \theta_s - \tilde{y}_\lambda), \qquad (H + \lambda I) w_q = \nabla_\theta f_q(\theta_s), \tag{8}$$

which is exact and needs one solve per query. The shift always applies to the training example's gradient; the query gradient is evaluated at θ_s.

A Lanczos–Taylor approximation of w_q. The Lanczos basis $V = [v_1, \ldots, v_k]$ started from g_s, with Ritz values $\theta_1, \ldots, \theta_k$, is already available from the λ-selection algorithm. With $g_q = \nabla_\theta f_q(\theta_s)$, we split g_q into its part in this subspace and a residual $r_q = g_q - VV^\top g_q$. We approximate w_q in three steps.

The in-subspace part is solved in closed form: $w_q^{\text{in}} = \sum_{j=1}^{k} \frac{v_j^\top g_q}{\theta_j + \lambda} v_j$. The residual is refined with a two-dimensional Rayleigh–Ritz step on $\text{span}\{r_q, Hr_q\}$, which costs one Hessian-vector product and gives Ritz pairs $(\theta_{r,j}, p_j)$. This yields $w_q^{\text{res}} = \sum_{j=1}^{2} \frac{p_j^\top r_q}{\theta_{r,j} + \lambda} p_j$ and $w_q^{\text{cheap}} = w_q^{\text{in}} + w_q^{\text{res}}$.

Viewing w_q as a function of λ, we extrapolate to a smaller λ_{target} by its second-order Taylor expansion,

$$w_q(\lambda_{\text{target}}) \approx w_q^{\text{cheap}} - (\lambda_{\text{target}} - \lambda)(H + \lambda I)^{-1} w_q^{\text{cheap}} + (\lambda_{\text{target}} - \lambda)^2 (H + \lambda I)^{-2} w_q^{\text{cheap}}, \tag{9}$$

using two plain MINRES solves at the already well-conditioned λ, never at λ_{target} itself.

The series converges within the distance to the nearest pole of $(H + \lambda I)^{-1}$, $R = \lambda - |\theta_{\min}|$. We set $\lambda_{\text{target}} - \lambda = -0.85R$, a fixed function of λ and $\theta_{\min}$ with no parameter fit to the evaluation metric. The Taylor step moves the estimate toward the solution at λ_{target}, which lets it use less damping than λ-selection permits for a direct solve while never solving near a pole (Appendix C).

When the approximation applies. The approximation is only as good as the subspace. The residual fraction $\rho = \frac{1}{N_{\text{query}}} \sum_q \|r_q\| / \|g_q\|$ is available before any scores are computed, since it needs only the projection onto V. When $\rho < 0.5$ we use the Lanczos–Taylor approximation; when $\rho \geq 0.5$, most of each query gradient lies outside the subspace, and we solve Equation 8 directly instead. The threshold 0.5 is an empirical choice (Appendix A.4).

4 Experiments

We test whether our influence estimator $\mathcal{I}_\lambda$ (§3) gives better training-data attributions than existing estimators at the non-converged checkpoints that training actually produces, measured by the linear datamodeling score (LDS).

4.1 Experimental setup

Settings and evaluation. We use three small-scale LDS settings of ASTRA (Wang et al., 2025): UCI Concrete (regression) and 6,144-example subsets of MNIST and FashionMNIST (10-class). Each uses a four-layer ReLU MLP (d between 3.4×10^4 and 5.7×10^5) trained with SGD for a fixed 20 epochs without early stopping, so the final checkpoint θ_s is in general not stationary ($g_s \neq 0$); details are in Appendix A.1. LDS (Park et al., 2023) measures how well additive group attributions predict the effect of retraining on random subsets. We draw $M = 100$ subsets of half the training set and retrain on each 100 times (Concrete) or 50 times (images), averaging the outcome. The measured quantity is the absolute error for regression and the correct-class margin for classification. LDS is the per-query Spearman correlation between predicted and retrained outcomes, averaged over

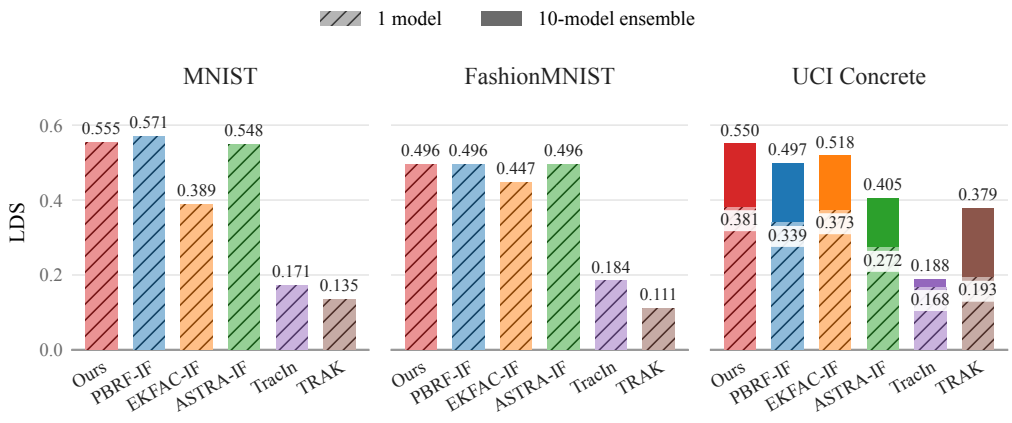

Figure 1: LDS (higher is better). Hatched bars: single model (MNIST and FashionMNIST: seed 0; Concrete: mean over 10 base seeds). Solid bars: 10-model ensemble. Ours uses the exact solve on MNIST and FashionMNIST and the Taylor approximation on Concrete. Numerical values are in Table 3.

queries. On Concrete we train 10 base models from independent seeds, report single-model LDS as the mean over them, and, following ASTRA, a 10-model ensemble: LDS of the average of the 10 score matrices. On MNIST and FashionMNIST we report a single model (seed 0).

Baselines. **PBRF-IF** computes the PBRF estimator $-(G+\lambda I)^{-1}\nabla_\theta L$ of Bae et al. (2022) directly, with the exact full-dataset Gauss–Newton matrix G and conjugate gradient. It is the special case of our estimator with H replaced by G and g_s set to zero (§3.2), here run at ASTRA's damping rather than at our λ. We also compare against **EKFAC-IF** (George et al., 2018; Grosse et al., 2023), **ASTRA-IF** (Wang et al., 2025), **TracIn** (Pruthi et al., 2020) and **TRAK** (Park et al., 2023). The IF baselines use ASTRA's per-setting damping. All methods are scored on the same checkpoints and ground truth. Solver settings and sign conventions are in Appendix A.2.

Our method. **Ours** follows the switching rule of the method. When the residual fraction ρ (the mean fraction of the query gradient outside the Lanczos subspace) is below 0.5, it uses the Lanczos–Taylor approximation; otherwise it solves System equation 5 exactly with MINRES and the Chebyshev preconditioner of §3.4. This selects the Taylor approximation on every Concrete seed ($\rho = 0.40$–0.50) and the exact solve on MNIST ($\rho = 0.60$) and FashionMNIST ($\rho = 0.61$). On Concrete, λ comes from the selection procedure of §3.3. On the image settings we instead set $\lambda = 1.4\,|\theta_{\min}|$, a 40% margin above the nearest pole, without the trust-region bound. Hyperparameters and this choice are detailed in Appendix A.3.

4.2 Results

UCI Concrete. Ours has the highest LDS both for a single model (0.381, against 0.373 for EKFAC-IF, 0.339 for PBRF-IF and 0.272 for ASTRA-IF) and for the 10-model ensemble (0.550, against 0.518 for EKFAC-IF and 0.497 for PBRF-IF; Figure 1). The single-model margin over EKFAC-IF is well within the seed-to-seed standard error (0.016), so we regard the two as comparable for a single model. The margins over PBRF-IF and ASTRA-IF, and all margins in the ensemble, are larger.

MNIST and FashionMNIST. On MNIST, Ours (0.555) is second to PBRF-IF (0.571), ahead of ASTRA-IF (0.548), and well ahead of EKFAC-IF (0.389), TracIn and TRAK. On FashionMNIST, Ours, PBRF-IF and ASTRA-IF tie at 0.496, ahead of EKFAC-IF (0.447), TracIn and TRAK. These are single-seed comparisons: on MNIST the seed-to-seed standard deviation of the IF baselines is about 0.02, comparable to the gaps between Ours, PBRF-IF and ASTRA-IF.

When the Taylor approximation applies. On Concrete, where $\rho \leq 0.50$ on every seed, the Taylor approximation outperforms the exact solve at the same λ (single model: 0.381 over 10 seeds vs. 0.283 over seeds 2–9, the seeds with an exact-solve run; ensemble: 0.550 vs. 0.395; Appendix A.5). It is not cheaper: the extrapolation adds two MINRES solves per query at λ, where the exact solve needs one. When $\rho \geq 0.5$, most of each query gradient lies outside the Lanczos subspace, which the two-dimensional refinement cannot represent, so the rule falls back to the exact solve. Because ρ is a projection onto an already-computed basis, it is available before any scores are computed, at no extra cost.

Scope. All experiments use MLPs with at most 5.7×10^5 parameters, and the image-classification results use a single seed. Multi-seed image results, further settings, convolutional and transformer models, and solver convergence are left to future work.

5 Conclusion

We generalize influence functions to non-stationary checkpoints without abandoning the true Hessian H. Our estimator recovers the classical formulation and the PBRF as special cases, while retaining the negative curvature that distinguishes a saddle point from a genuine minimum, information the Gauss-Newton Hessian G cannot represent. A λ-selection algorithm keeps $H + \lambda I$ invertible while balancing this curvature preservation against the validity of the underlying quadratic approximation, and a MINRES solver, together with a Lanczos–Taylor approximation that reuses the Krylov basis from λ-selection, computes both linear systems using only Hessian-vector products.

On UCI Concrete, our estimator attains the highest LDS for a 10-model ensemble (0.550, against 0.518 for EKFAC-IF) and is comparable to EKFAC-IF for a single model, ahead of PBRF-IF and ASTRA-IF. On MNIST, where the residual-fraction test selects the direct solve, it is second to PBRF-IF and ahead of ASTRA-IF and EKFAC-IF. On FashionMNIST, also with the direct solve, it ties PBRF-IF and ASTRA-IF at 0.496. These comparisons are limited to MLPs with at most 5.7×10^5 parameters, and the image results use a single seed.

Two directions follow directly. First, the residual-fraction threshold that triggers our fallback is set empirically; a checkpoint-derived criterion, tied to the same spectral quantities already computed for λ-selection, could replace it. Second, our results are limited to models small enough to admit repeated retraining for ground-truth comparison; extending this evaluation to checkpoints of a large language model, where non-convergence is the rule rather than the exception, is the natural next test of whether preserving H's curvature matters as much in practice as our theory suggests it should.

AI use statement

In this work, we used generative AI tools (large language model assistants) only for tasks in the recommended-disclosure categories of the ICLR 2027 AI Policy for Authors. First, we used them to draft parts of the paper text, which the authors then revised. Second, we used them to edit the paper for readability and concision. Third, we used them to identify and search for relevant literature and to format references. We have not used generative AI tools to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, or to prove them or write their proofs; translation, dataset cleaning and reformatting, and qualitative data analysis are not applicable to this work. The authors reviewed all AI-assisted text and revised it where needed, checked every AI-suggested reference against the original publication, and checked all reported numbers against the experiment outputs. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18:4148–4187, 2017.

Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger Grosse. If influence functions are the answer, then what is the question? In *Advances in Neural Information Processing Systems*, volume 35, pp. 17953–17967, 2022. URL

https://proceedings.neurips.cc/paper_files/paper/2022/hash/7234e0c36fdbcb23e7bd56b68838999b-Abstract-Conference.html.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolling. In *Advances in Neural Information Processing Systems*, volume 37, pp. 66647–66686, 2024.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1171–1179, 2015.

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pp. 437–478. Springer, Berlin, Heidelberg, 2nd edition, 2012. doi: 10.1007/978-3-642-35289-8_26.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018. doi: 10.1137/16M1080173.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. Understanding the origins of bias in word embeddings. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 803–811. PMLR, 2019.

Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. MPS-SIAM Series on Optimization. SIAM, 2000. doi: 10.1137/1.9780898719857.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, volume 27, pp. 2933–2941, 2014.

Tobin A. Driscoll and Richard J. Braun. *Fundamentals of Numerical Computation: Julia Edition*. SIAM, 2022. doi: 10.1137/1.9781611977011.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In *Advances in Neural Information Processing Systems*, volume 31, pp. 9550–9560, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/48000647b315f6f00f913caa757a70b3-Abstract.html.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2242–2251. PMLR, 2019.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Nicholas I. M. Gould, Stefano Lucidi, Massimo Roma, and Philippe L. Toint. Solving the trust-region subproblem using the lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999. doi: 10.1137/S1052623497322735.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023. URL https://arxiv.org/abs/2308.03296.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5553–5563, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.492.

Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162. PMLR, 2022. URL https://proceedings.mlr.press/v162/ilyas22a.html.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 2017. URL https://proceedings.mlr.press/v70/koh17a.html.

Shuming Kong, Yanyan Shen, and Linpeng Huang. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=EskfH0bwNVn.

Donghoon Lee, Hyunsin Park, Trung Pham, and Chang D. Yoo. Learning augmentation network via influence functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10961–10970, 2020.

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

Jorge J. Moré and Danny C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983. doi: 10.1137/0904038.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A. Raffel. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Christopher C. Paige. Error analysis of the lanczos algorithm for tridiagonalizing a symmetric matrix. *IMA Journal of Applied Mathematics*, 18(3):341–349, 1976. doi: 10.1093/imamat/18.3.341.

Christopher C. Paige and Michael A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975. doi: 10.1137/0712047.

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5012–5021. PMLR, 2019.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27074–27113. PMLR, 2023. URL https://proceedings.mlr.press/v202/park23c.html.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19920–19930, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e6385d39ec9394f2f3a354d9d2b88eec-Abstract.html.

Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Peter Schulam and Suchi Saria. Can you trust this prediction? auditing pointwise reliability after learning. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1022–1031. PMLR, 2019.

Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, extended edition, 2019.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of $\mathrm{tr}(f(a))$ via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017. doi: 10.1137/16M1104974.

Eugene Vecharynski and Andrew V. Knyazev. Absolute value preconditioning for symmetric indefinite linear systems. *SIAM Journal on Scientific Computing*, 35(2):A696–A718, 2013. doi: 10.1137/120886686.

Andrew Wang, Elisa Nguyen, Runshi Yang, Juhan Bae, Sheila A. McIlraith, and Roger Grosse. Better training data attribution via better inverse hessian-vector products. *Advances in Neural Information Processing Systems*, 2025. URL `https://arxiv.org/abs/2507.14740`. arXiv:2507.14740.

A Experimental details

A.1 Settings and training

Table 1: Experimental settings. d is the number of parameters. All models are four-layer ReLU MLPs trained in float64 for 20 epochs with SGD (momentum 0.9) at a constant learning rate (LR) and weight decay (WD). Ground-truth retrains are per base seed (subsets $\times$ repeats).

Setting	Task	Hidden widths	N_{train}	N_{query}	d	LR	WD	Batch	GT retrains
Concrete	regression	128–128–128	896	103	34,305	3×10^{-2}	10^{-5}	32	100×100
MNIST	10-class	512–256–128	6,144	100	567,434	3×10^{-2}	10^{-3}	64	100×50
FMNIST	10-class	512–256–128	6,144	100	567,434	3×10^{-2}	10^{-3}	64	100×50

LDS measurement. For regression the measured quantity is $|f(x) - y|$. For classification it is the correct-class margin $-g_t + \log \sum_{i \neq t} \exp(g_i)$, where g are the logits and t is the true class. Each subset contains a fraction $\beta = 0.5$ of the training set. On Concrete, standard errors are $s/\sqrt{n}$ over the 10 base seeds, and the 10-model ensemble is scored against a separate ground-truth set generated with the same protocol. On MNIST and FashionMNIST all methods are scored on the seed-0 base model.

A.2 Baselines

Table 2: Baseline settings. The damping of the IF baselines follows ASTRA's per-setting values.

Setting	IF damping	ASTRA-IF iterations	TRAK proj. dim	TracIn checkpoints
Concrete	0.0017	200	512	5
MNIST	0.0052	200	4096	5
FMNIST	0.0052	200	4096	5

- **PBRF-IF** computes $-(G + \lambda I)^{-1} \nabla_\theta L$ with the exact Gauss–Newton matrix over the full training set, solved by unpreconditioned conjugate gradient to relative tolerance 10^{-6}.
- **EKFAC-IF** uses the `kronfluence` implementation.
- **ASTRA-IF** uses EKFAC-preconditioned stochastic Neumann iterations with inner learning rate 0.1λ, batch size 256, and the learning rate halved every 50 iterations.
- **TracIn** uses 5 evenly spaced checkpoints from the same training run, including the final one.
- **TRAK** uses a single model per seed, matching every other method, rather than TRAK's usual ensemble.

Following the reference implementations' sign conventions, EKFAC-IF, TracIn and, on the classification settings, TRAK are sign-aligned with the measurement. Without this alignment TRAK's LDS on MNIST is about -0.12. For regression, TRAK featurizes the raw prediction $g(\theta, x_q)$, whereas the measurement is $|g(\theta, x_q) - y_q|$. The two gradients differ by $\operatorname{sign}(g(\theta, x_q) - y_q)$, so we multiply each query's TRAK scores by $-\operatorname{sign}(g(\theta_s, x_q) - y_q)$. Without this correction TRAK's LDS on Concrete is -0.001 (single model) and -0.026 (ensemble).

A.3 Ours: exact solve

Ours computes each per-query solve with the Lanczos–Taylor approximation (Appendix A.4) when the residual fraction $\rho < 0.50$, and with the exact solve described here otherwise. The exact solve runs MINRES to relative tolerance 10^{-6} with the Chebyshev preconditioner of §3.4, using degree $q = 80$ and cap $\lambda_{\text{cap}} = 10\lambda$.

Concrete. λ is selected as in §3.3. θ_{ref} comes from stochastic Lanczos quadrature (Ubaru et al., 2017) with 20 random probes of depth 20, taking the largest gap between consecutive sorted eigenvalue estimates that has at least 10 estimates on the outlier side. When $\lambda_{\text{cluster}} = \tau_{\text{safe}} + |\theta_{\min}|$ exceeds $\lambda_{\max}$, $\varepsilon_{\text{damp}}$ is relaxed only as far as needed, which was necessary on every seed. On seeds 2–9, the selected λ lies between 1.19 and 3.30, with $\theta_{\min}$ between -0.56 and -0.19. We use $\tau_{\text{safe}} = 1$ and $\Delta_{\max} = (6\tau_{\text{model}}/L_3)^{1/3}$ with τ_{model} proportional to $\mathcal{J}(\theta_s)$. The training gradients are evaluated at $\theta_s - \tilde{y}_\lambda$, where, with $A = H + \lambda I$, $c = A^{-1} g_s$ (the System equation 4 solution) and $z_1 = -A^{-1}c$, $\tilde{y}_\lambda = \Delta_{\max}(c + \beta_1 z_1)/\|c + \beta_1 z_1\|$. For $\beta_1 = 0$ this is the System equation 4 solution rescaled to the trust radius. On Concrete the exact solve is an ablation; Ours uses the Taylor approximation there.

MNIST and FashionMNIST. Ours uses the exact solve ($\rho \geq 0.5$) at $\lambda = 1.4\,|\theta_{\min}|$, i.e. λ_{cluster} with $\tau_{\text{safe}} = 0.4\,|\theta_{\min}|$ ($\lambda = 0.024$ on MNIST seed 0, where $\theta_{\min} = -0.0172$; on FashionMNIST seed 0, $\theta_{\min} = -0.385$). This rule was chosen from two values tried on MNIST seed 0 ($1.4\,|\theta_{\min}|$ and 0.02, with LDS 0.555 and 0.550). The trust-region bound λ_{trust} is not applied, and the shift is the System equation 4 solution itself, y_λ, without rescaling.

A.4 Ours: Lanczos–Taylor approximation and the switching rule

The Taylor approximation keeps the λ and evaluation point of the exact solve and approximates w_q in equation 8. Let $V = [v_1, \ldots, v_k]$ be the Lanczos basis started from g_s that λ-selection already computes, with Ritz values $\theta_1, \ldots, \theta_k$. The depth k is chosen automatically; $k = 80$ on Concrete. Write $g_q = \nabla_\theta f_q(\theta_s)$.

1. **In-subspace term.** $w_q^{\text{in}} = \sum_{j=1}^{k} \frac{v_j^\top g_q}{\theta_j + \lambda}\, v_j$.
2. **Out-of-subspace refinement.** Let $r = g_q - VV^\top g_q$ be the part of g_q outside the basis. A Rayleigh–Ritz step on $\operatorname{span}\{r, Hr\}$ (one extra Hessian-vector product) gives Ritz pairs $(\theta_{r,j}, p_j)_{j=1,2}$, and $w_q^{\text{res}} = \sum_{j=1}^{2} \frac{p_j^\top r}{\theta_{r,j} + \lambda}\, p_j$. We set $w_q^{\text{cheap}} = w_q^{\text{in}} + w_q^{\text{res}}$.
3. **Extrapolation in λ.** Let $R = \lambda - |\theta_{\min}|$ be the distance from λ to the nearest pole of $(H + \lambda I)^{-1}$, and $\beta = -0.85\,R$, so that $\lambda_{\text{target}} = \lambda + \beta$ stays inside the radius of convergence. The final estimate is the second-order Taylor expansion $w_q \propto w_q^{\text{cheap}} - \beta A^{-1} w_q^{\text{cheap}} + \beta^2 A^{-2} w_q^{\text{cheap}}$, computed with two unpreconditioned MINRES solves at λ and rescaled to the norm of w_q^{cheap}. The rescaling does not affect per-query rankings.

Residual fraction. $\rho = \frac{1}{N_{\text{query}}} \sum_q \|r_q\|/\|g_q\|$ is the mean fraction of the query gradient that lies outside the Lanczos basis. It needs only the projection in step 2, so it is available before any scores are computed. Ours uses the Taylor approximation when $\rho < 0.50$ and the exact solve otherwise. The 0.50 threshold is an empirical choice made on preliminary runs and has not been validated beyond the settings reported here. On Concrete seed 9, $\rho = 0.49998$, marginally below the threshold.

Per-setting hyperparameters. The safety margin τ_{safe} is an empirical, per-task hyperparameter: $\tau_{\text{safe}} = 1$ on Concrete and $\tau_{\text{safe}} = 0.4\,|\theta_{\min}|$ on MNIST and FashionMNIST (see above). The shift parameter is $\beta_1 = 0$ on Concrete, except seed 1, where $\beta_1 = 4$ was selected on that seed's LDS.

A.5 LDS RESULTS

Table 3: LDS values plotted in Figure 1. **(a)** Single model: MNIST and FashionMNIST seed 0; Concrete mean over 10 base seeds. **(b)** 10-model ensemble (Concrete only). **(c)** The two components of Ours on Concrete, single model [10-model ensemble in brackets]; Ours uses the Taylor approximation there. †Mean over seeds 2–9.

Method	MNIST (seed 0)	FMNIST (seed 0)	Concrete
(a) Single model			
PBRF-IF	0.571	0.496	0.339
EKFAC-IF	0.389	0.447	0.373
ASTRA-IF	0.548	0.496	0.272
TracIn	0.171	0.184	0.168
TRAK	0.135	0.111	0.193
Ours	0.555	0.496	0.381
(b) 10-model ensemble			
PBRF-IF	–	–	0.497
EKFAC-IF	–	–	0.518
ASTRA-IF	–	–	0.405
TracIn	–	–	0.188
TRAK	–	–	0.379
Ours	–	–	0.550
(c) Components of Ours on Concrete			
Exact solve	–	–	$0.283^{\dagger}$ [0.395]
Taylor approximation	–	–	0.381 [0.550]

A.6 SOLVER COST

We measure MINRES on the UCI Concrete seed-0 checkpoint ($d = 34{,}305$) at the λ selected by §3.3 ($\lambda = 1.343$, $\theta_{\min} = -0.215$) and at two smaller λ closer to the pole, with and without the Chebyshev preconditioner ($q = 80$, $\lambda_{\text{cap}} = 10\lambda$). Each solve targets relative tolerance 10^{-6} for $(H + \lambda I)w = \nabla_\theta f_q(\theta_s)$, for the first 5 queries. We count Hessian-vector products (HVPs) inside the operator, so the preconditioner's own HVPs are included, and report the true relative residual $\|\nabla f_q - (H + \lambda I)w\|/\|\nabla f_q\|$ at termination (Table 4).

Table 4: MINRES cost on UCI Concrete (seed 0), mean over 5 queries; ranges in brackets. Building the preconditioner costs a further 480 HVPs per λ.

λ ($\theta_{\min} + \lambda$)	Preconditioner	Iterations	HVPs per solve	True relative residual
1.343 (1.13), selected	none	12.6 [12–13]	13	3×10^{-6}–9×10^{-6}
	Chebyshev	13.6 [13–14]	1,182	7×10^{-7}–3×10^{-6}
0.265 (0.05)	none	31.0 [30–32]	31	3×10^{-5}–5×10^{-5}
	Chebyshev	29.4 [29–30]	2,461	6×10^{-6}–1×10^{-5}
0.225 (0.01)	none	53.0 [52–55]	53	5×10^{-5}–8×10^{-5}
	Chebyshev	48.2 [47–49]	3,984	1×10^{-5}–2×10^{-5}

At the selected λ, plain MINRES converges in about 13 HVPs. The preconditioner does not reduce the iteration count there and multiplies the cost by about $90\times$: with $\lambda_{\text{cap}} = 10\lambda$ above most of the spectrum, the fitted polynomial is nearly constant and acts as a scaled identity. Closer to the pole, iterations grow (31 and 53), and the preconditioner reduces them by 5–9% while reaching a lower true residual. Without a preconditioner, MINRES's internal stopping test terminates with a true residual above the 10^{-6} target, so matching the preconditioned residual would take a few more plain iterations. The Lanczos–Taylor approximation uses plain MINRES for its two solves at λ.

A.7 COMPUTE

Ground-truth retraining dominates the cost of the evaluation. With the Chebyshev preconditioner, the exact solve costs 6.5 s per query on Concrete and 90–102 s on MNIST/FMNIST (one CPU node, $q = 80$, relative tolerance 10^{-6}). On a V100 GPU, an MNIST exact solve took about 37 s per query, measured at $\lambda = 0.02$ rather than the reported $\lambda = 0.024$, against about 15 s per query for ASTRA-IF's 200 Neumann iterations on the same GPU. The per-seed wall-clock times of the Concrete runs (1.4–2.6 h) include ground-truth generation. The Taylor approximation shares the k-step Lanczos basis across queries, and needs one Hessian-vector product and two unpreconditioned MINRES solves per query at λ.

B THE λ-SELECTION ALGORITHM, IN FULL

Given a checkpoint θ_s and its gradient g_s, this appendix describes how we compute $\lambda_{\max}$, $\lambda_{\min}$, and the final λ used in Systems equation 4 and equation 5, expanding on the compressed presentation in §3.3.

Step 1: Estimating θ_{ref} (for $\lambda_{\max}$). We estimate H's spectrum by stochastic Lanczos quadrature (Ubaru et al., 2017): short Lanczos runs from several independent random probes, pooled. A single Lanczos run is not enough, since it misses any outlier whose eigenvector happens to have small overlap with its starting vector. We take the largest ratio between consecutive sorted eigenvalue estimates as the boundary θ_{ref}, considering only boundaries backed by a minimum number of estimates on the outlier side, since a ratio between two near-zero estimates can otherwise dominate spuriously. This gives $\lambda_{\max} = \theta_{\text{ref}} \cdot \varepsilon_{\text{damp}}/(1 - \varepsilon_{\text{damp}})$.

Step 2: Estimating $\theta_{\min}$ (for λ_{cluster}). We run Lanczos until its Ritz residual meets the solver tolerance, rather than for a fixed number of steps; a fixed budget can underestimate $|\theta_{\min}|$ substantially, since the most negative eigenvalue is often the slowest to converge. This gives $\lambda_{\text{cluster}} = \tau_{\text{safe}} + |\theta_{\min}|$.

Step 3: Estimating L_3 (for λ_{trust}). We estimate L_3 by comparing Hessian-vector products at θ_s and at $\theta_s - y$ along a provisional System equation 4 step y computed at $\lambda_{\max}$, costing two Hessian-vector products. This gives $\Delta_{\max} = (6\tau_{\text{model}}/L_3)^{1/3}$.

Step 4: Solving for λ_{trust} via the secular equation. With $g_s = \sum_i c_i^{(0)} v_i$ in H's eigenbasis, $\|y_\lambda\|^2 = \sum_i (c_i^{(0)})^2/(\theta_i + \lambda)^2$ is a closed-form function of λ alone, computable at any λ from a single Lanczos process started from g_s, without further Hessian-vector products (Gould et al., 1999; Moré & Sorensen, 1983). This function is monotonically decreasing in λ for $\lambda > \lambda_{\text{cluster}}$, so we find the root of $\|y_\lambda\|^2 - \Delta_{\max}^2 = 0$ by bisection on this interval, giving λ_{trust}.

Step 5: The noise floor. Gradient noise of scale ε_g (from minibatch estimation of g_s; Bottou et al., 2018) is amplified by at most ε_g/λ, so a tolerance ε_{tol} gives $\lambda_{\text{floor}} = \varepsilon_g/\varepsilon_{\text{tol}}$.

Step 6: Combining the bounds, and the relaxation guard. Set $\lambda_{\min} = \max(\lambda_{\text{floor}}, \lambda_{\text{cluster}}, \lambda_{\text{trust}})$. If $\lambda_{\min} \le \lambda_{\max}$, any $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ satisfies every requirement, and we use $\lambda_{\min}$. If λ_{cluster} or λ_{trust} pushes $\lambda_{\min}$ above $\lambda_{\max}$, clipping to $\lambda_{\max}$ would proceed at an unsafe λ that violates the cluster or trust-region guard. Instead, we relax $\varepsilon_{\text{damp}}$, exactly as far as needed for $\lambda_{\max}$ to reach $\lambda_{\min}$, with no fixed relaxation ceiling. Since $\|y_\lambda\|$ decreases toward zero as λ grows, λ_{trust} is always finite, so a λ satisfying every requirement always exists. We report the relaxed $\varepsilon_{\text{damp}}$ as a per-checkpoint diagnostic of how much curvature had to be given up to remain safe.

C PRECONDITIONED MINRES AND THE LANCZOS–TAYLOR APPROXIMATION, IN FULL

This appendix expands on the compressed presentation in §3.4: the construction of the Chebyshev-fitted preconditioner and its convergence guarantee, and the derivation and validation of the Lanczos–Taylor approximation of w_q, including the target λ_{target}.

Algorithm 1 λ-selection

Require: checkpoint θ_s, gradient g_s, tolerances τ_{safe}, τ_{model}, ε_{tol}, $\varepsilon_{\text{damp}}$
Ensure: selected λ, relaxed $\varepsilon_{\text{damp}}$ (if applicable)
Estimate θ_{ref} via pooled stochastic Lanczos quadrature (Step 1)
$\lambda_{\max} \leftarrow \theta_{\text{ref}} \cdot \varepsilon_{\text{damp}}/(1 - \varepsilon_{\text{damp}})$
Estimate $\theta_{\min}$ via Lanczos run to convergence (Step 2)
$\lambda_{\text{cluster}} \leftarrow \tau_{\text{safe}} + |\theta_{\min}|$
Estimate L_3 via two Hessian-vector products at $\lambda_{\max}$ (Step 3)
$\Delta_{\max} \leftarrow (6\tau_{\text{model}}/L_3)^{1/3}$
Run Lanczos from g_s; solve secular equation for λ_{trust} via bisection (Step 4)
$\lambda_{\text{floor}} \leftarrow \varepsilon_g/\varepsilon_{\text{tol}}$ (Step 5)
$\lambda_{\min} \leftarrow \max(\lambda_{\text{floor}}, \lambda_{\text{cluster}}, \lambda_{\text{trust}})$
if $\lambda_{\min} \leq \lambda_{\max}$ **then**
    return $\lambda_{\min}$, $\varepsilon_{\text{damp}}$ (unrelaxed)
else
    relax $\varepsilon_{\text{damp}}$ until $\lambda_{\max} = \lambda_{\min}$
    return $\lambda_{\min}$, relaxed $\varepsilon_{\text{damp}}$
end if

Construction of the Chebyshev-fitted preconditioner. We estimate $H + \lambda I$'s extreme eigenvalues, $\nu_{\min}$ and $\nu_{\max}$, via a short power-iteration pass. We fit a degree-q polynomial p at Chebyshev nodes on $[\nu_{\min}, \nu_{\max}]$ (Trefethen, 2019). Chebyshev nodes cover the whole interval by construction, which avoids the oscillation and extrapolation failures of fitting at irregular sample locations such as Lanczos Ritz values. Since $1/|\nu|$ is singular at zero, we cap the denominator at $\lambda_{\text{cap}} = 10\lambda$, fitting

$$p(\nu) \approx \frac{1}{\max(|\nu|, \lambda_{\text{cap}})} \tag{10}$$

instead. Tying the cap to λ makes it checkpoint-adaptive. The cap changes only the preconditioner, never the system being solved. We apply P^{-1} via the stable Chebyshev three-term recurrence, costing q Hessian-vector products per use, so each MINRES iteration costs $q + 1$ in total. We use $q = 80$; doubling it to 160 gave nearly identical convergence.

Convergence guarantee. Let α denote the worst-case deviation of the preconditioned eigenvalues $p(\nu_i)\nu_i$ from ± 1, for eigenvalues with $|\nu_i| \geq \lambda_{\text{cap}}$:

$$\alpha = \max_{|\nu_i| \geq \lambda_{\text{cap}}} \big| p(\nu_i)\nu_i - \operatorname{sign}(\nu_i) \big|. \tag{11}$$

MINRES then satisfies

$$\|r_m\| \leq \alpha^{\lfloor m/2 \rfloor} \|r_0\|. \tag{12}$$

The residual decomposition, restated. Recall from §3.4 that we approximate $w_q = (H + \lambda I)^{-1} g_q$ by splitting g_q using the Lanczos basis $V = [v_1, \ldots, v_k]$, with Ritz values $\theta_1, \ldots, \theta_k$, into

$$g_q = V V^\top g_q + r_q, \qquad r_q = g_q - V V^\top g_q. \tag{13}$$

The in-subspace part is solved in closed form,

$$w_q^{\text{in}} = \sum_{j=1}^{k} \frac{v_j^\top g_q}{\theta_j + \lambda} \, v_j. \tag{14}$$

The residual r_q is refined with a two-dimensional Rayleigh–Ritz step on $\operatorname{span}\{r_q, H r_q\}$, which costs one Hessian-vector product and gives Ritz pairs $(\theta_{r,j}, p_j)$ for $j = 1, 2$. This yields

$$w_q^{\text{res}} = \sum_{j=1}^{2} \frac{p_j^\top r_q}{\theta_{r,j} + \lambda} \, p_j, \qquad w_q^{\text{cheap}} = w_q^{\text{in}} + w_q^{\text{res}}. \tag{15}$$

Deriving the Taylor extrapolation. Viewing w_q as a function of λ evaluated at g_q, write $A(\lambda) = H + \lambda I$, so $w_q(\lambda) = A(\lambda)^{-1} g_q$. Differentiating with respect to λ,

$$\frac{dw_q}{d\lambda} = -A(\lambda)^{-1} \frac{dA}{d\lambda} A(\lambda)^{-1} g_q = -A(\lambda)^{-1} w_q(\lambda), \tag{16}$$

since $dA/d\lambda = I$. Differentiating a second time,

$$\frac{d^2 w_q}{d\lambda^2} = \frac{d}{d\lambda}\Big[- A(\lambda)^{-1} w_q(\lambda)\Big] = A(\lambda)^{-2} w_q(\lambda) - A(\lambda)^{-1} \frac{dw_q}{d\lambda} = 2A(\lambda)^{-2} w_q(\lambda). \tag{17}$$

The second-order Taylor expansion of $w_q(\lambda_{\text{target}})$ around λ, using Equations 16 and 17 and substituting w_q^{cheap} for $w_q(\lambda)$, is

$$w_q(\lambda_{\text{target}}) \approx w_q^{\text{cheap}} - (\lambda_{\text{target}} - \lambda)(H + \lambda I)^{-1} w_q^{\text{cheap}} + (\lambda_{\text{target}} - \lambda)^2 (H + \lambda I)^{-2} w_q^{\text{cheap}}, \tag{18}$$

which requires two additional MINRES solves at the already well-conditioned λ: one with right-hand side w_q^{cheap} to obtain $(H + \lambda I)^{-1} w_q^{\text{cheap}}$, and one more with that result as the new right-hand side to obtain $(H + \lambda I)^{-2} w_q^{\text{cheap}}$.

Radius of convergence and choosing λ_{target}. Writing $w_q(\lambda)$ in H's eigenbasis, its i-th coefficient is $g_i/(\theta_i + \lambda)$, which has a pole at $\lambda = -\theta_i$. The nearest such pole to λ is at $\lambda = -\theta_{\min}$, so the Taylor series in Equation 18 converges for

$$|\lambda_{\text{target}} - \lambda| < R, \qquad R \equiv \lambda - |\theta_{\min}|. \tag{19}$$

We set

$$\lambda_{\text{target}} - \lambda = -0.85\, R, \tag{20}$$

a fixed function of λ and $\theta_{\min}$ with no parameter fit to the evaluation metric.

Accuracy of the expansion. In H's eigenbasis the i-th coefficient of w_q is scaled by $1/(1 - t_i) = 1 + t_i + t_i^2 + \cdots$ with $t_i = -(\lambda_{\text{target}} - \lambda)/(\theta_i + \lambda)$, and Equation 18 keeps the first three terms. Directions with large θ_i ($t_i \approx 0$) are reproduced almost exactly; the direction of $\theta_{\min}$ has $t = 0.85$, where the truncation keeps 2.57 of the exact factor 6.67. The Taylor step therefore moves the estimate toward, but not fully to, the solution at λ_{target}, which lets it use less damping than λ-selection permits for a direct solve while never solving near a pole.