Better Hessians Matter: Studying the Impact of Curvature Approximations in Influence Functions

Steve Hong
University of Cambridge
mdh58@cam.ac.uk

Runa Eschenhagen
University of Cambridge
re393@cam.ac.uk

Bruno Mlodozeniec
University of Cambridge
bkm28@cam.ac.uk

Richard E. Turner
University of Cambridge
Alan Turing Institute
ret26@cam.ac.uk

Abstract

Influence functions offer a principled way to trace model predictions back to training data, but their use in deep learning is hampered by the need to invert a large, ill-conditioned Hessian matrix. Approximations such as Generalised Gauss-Newton (GGN) and Kronecker-Factored Approximate Curvature (K-FAC) have been proposed to make influence computation tractable, yet it remains unclear how the departure from exactness impacts data attribution performance. Critically, given the restricted regime in which influence functions are derived, it's not necessarily clear better Hessian approximations should even lead to better data attribution performance. In this paper, we investigate the effect of Hessian approximation quality on influence-function attributions in a controlled classification setting. Our experiments show that *better Hessian approximations consistently yield better influence score quality*, offering justification for recent research efforts towards that end. We further decompose the approximation steps for recent Hessian approximation methods and evaluate each step's influence on attribution accuracy. Notably, the mismatch between K-FAC eigenvalues and GGN/EK-FAC eigenvalues accounts for the majority of the error and influence loss. These findings highlight which approximations are most critical, guiding future efforts to balance computational tractability and attribution accuracy.

1 Introduction

When attempting to understand the behaviour of a machine learning model, a common question is: how did the training examples contribute to a given model output? Which examples contributed the most? This can also be framed counterfactually: how would the predictions change if certain training examples were removed and the model was retrained? The goal of training–data attribution (TDA) methods [1] is to answer this question in a principled way.

Among these methods, influence functions [2–6] provide an efficient tool by exploiting the local structure of the loss landscape around the learned parameters. The efficiency of influence functions makes them attractive: given per-sample gradients and second-order curvature information, they approximate the effect of removing a training data point *without retraining*. Influence functions have been applied successfully in large–scale deep learning. For example, they have been used in 50 billion parameter Large Language Models (LLMs) to study generalisation [3], and in scalable data attribution for diffusion models [7].

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability.

A key practical challenge in influence function implemenetation is the Hessian bottleneck [2]. Exact computation of the inverse Hessian-vector product is intractable for modern models because the Hessian is often large and ill-conditioned [3, 8]. To address this, two broad approximation regimes are used. Iterative methods such as conjugate gradient [9] or LiSSA [10] approximate inverse Hessian–vector products by using an iterative solver. These methods are asymptotically exact, but often require thousands of steps for decent performance [2]. Structured approximations, on the other hand, replace the Hessian with stable and light-weight alternatives: the Generalised Gauss–Newton (GGN) [11], block–diagonal forms [12], and Kronecker–factored variants such as Kronecker-Factored Approximate Curvature (K-FAC) [13], usually with a separate eigenvalue correction step (EK-FAC) [14] to improve spectral fidelity. However, some of the approximations K-FAC and EK-FAC make are quite specific to the optimisation setting, in which they have been shown to have other desirable properties that lead to good down-stream performance, beyond the original goal of being tractable [15, 16].

Considerable effort in both the optimisation and data attribution communities has recently gone into developing more faithful curvature approximations [14, 17–20]. However, it is not obvious whether such efforts are beneficial for influence estimation: influence functions may be robust to some approximation errors, while they can be substantially sensitive to certain curvature information. Clarifying when and by how much better Hessian approximations improve influence function-based attribution would help determine whether the community should continue to invest in developing higher–fidelity curvature models, or whether the gains are marginal relative to their cost.

Core contributions. This work first decomposes the three approximation layers of K-FAC and examines the literature on when each approximation holds exactly versus when it introduces error. We then design controlled experiments to empirically investigate three questions:

1. Does higher-fidelity Hessian approximation improve influence scores?
2. Which approximation layer contributes most to the error, and what causes it?
3. Which approximation error is influence fidelity most sensitive to?

2 Related work

Fragility of influence functions. Basu et al. [21] show that influence estimates can misalign with leave-one-out retraining and are sensitive to model depth, regularisation, and query choice. However, Epifano et al. [22] attribute part of the effect to evaluation design and claim that regularisation alone is insufficient. Similarly, Mlodozeniec et al. [23] show that leave-one-out error is in large part attributable to stochastic initialisation and training, and suggest alternative evaluation and influence formulations that take that into account. Bae et al. [8] isolates warm starts, damping/proximity, non-convergence, linearisation, and solver terms, and argues that practical estimates often resemble a Proximal Bregman response function; solver-induced error remains underexplored. Group deletions show high rank correlation but possibly large absolute errors, clarifying when correlation is informative [24]. Ye et al. [25] propose an alternative influence function formulation that leverages flat validation minima to improve robustness. Recent work studies the LiSSA and EK-FAC approximation error with a focus on mislabel detection [26]. To our knowledge, the relationship between curvature-approximation error (Hessian$\rightarrow$GGN, block-diagonal, K-FAC/EK-FAC) and attribution quality across training regimes, depths, and widths has not been quantified. We provide a systematic evaluation in this work.

Hessian approximations for influence functions. Early implementations used iterative IHVP solvers, notably LiSSA [2, 10]. At larger scales, EK-FAC has been used to make influence estimates tractable [3, 14]. Two directions follow: faster and more stable iterative solvers, and higher-fidelity structured curvature (e.g., GGN/K-FAC variants). ASTRA [20] combines EK-FAC preconditioning with stochastic Neumann iterations to approximate damped-GGN iHVPs; relative to block-diagonal EK-FAC estimators it reduces iterations and improves attribution accuracy across architectures. These results motivate our controlled study of how GGN substitution, block-diagonality, and Kronecker factorisation trade off computational cost and attribution fidelity.

3 Background

We first establish the mathematical framework for influence functions and then detail the approximation layers that make them computationally tractable.

3.1 Data attribution with influence functions

Consider a dataset $\mathcal{D} = \{z_i\}_{i=1}^N$ where each $z_i = (x_i, y_i)$ represents an input–output pair in supervised learning; here, $x_i \in \mathbb{R}^{d_x}$ and $y_i \in \mathbb{R}^{d_y}$. We fit parameters $\theta^\star \in \mathbb{R}^D$ by minimising the empirical risk:

$$\theta^\star \;:=\; \arg\min_{\theta\in\mathbb{R}^D} J(\theta) \;=\; \arg\min_{\theta\in\mathbb{R}^D} \frac{1}{N}\sum_{i=1}^N L(z_i,\theta). \tag{1}$$

We evaluate model behaviour at a query z_q with a measurement $m(z_q, \theta)$ (e.g., a loss or score). For a training point $z_m \in \mathcal{D}$, an attribution method $\tau(z_q, z_m, \mathcal{D})$ quantifies how z_m affects $m(z_q, \theta^\star)$. To study this effect, introduce a scalar ϵ that up- or down-weights z_m and define the response function

$$r(\epsilon) \;:=\; \arg\min_{\theta\in\mathbb{R}^D} \; J(\theta) + \tfrac{\epsilon}{N}\, L(z_m,\theta), \tag{2}$$

with $\theta^\star := r(0)$ and $\mathbf{H} := \nabla_\theta^2 J(\theta^\star)$.

The associated first-order stationarity condition along the path $\epsilon \mapsto r(\epsilon)$ is

$$0 \;=\; \nabla_\theta J(r(\epsilon)) \;+\; \tfrac{\epsilon}{N}\,\nabla_\theta L(z_m, r(\epsilon)). \tag{3}$$

Differentiating this identity with respect to ϵ and evaluating at $(\theta^\star, 0)$ yields

$$\left.\frac{dr}{d\epsilon}\right|_{\epsilon=0} = -\,\mathbf{H}^{-1}\,\frac{1}{N}\,\nabla_\theta L(z_m,\theta^\star), \qquad r(\epsilon) \;\approx\; \theta^\star - \epsilon\,\mathbf{H}^{-1}\,\frac{1}{N}\,\nabla_\theta L(z_m,\theta^\star). \tag{4}$$

Setting $\epsilon = -1$ corresponds to removing z_m from the objective and gives the first-order parameter change

$$\theta^\star(\mathcal{D}\setminus\{z_m\}) - \theta^\star \;\approx\; \frac{1}{N}\,\mathbf{H}^{-1}\,\nabla_\theta L(z_m,\theta^\star). \tag{5}$$

Applying the chain rule to the query metric then yields the classical influence function

$$\tau_{\text{IF}}(z_q, z_m, \mathcal{D}) \;:=\; \nabla_\theta m(z_q,\theta^\star)^\top\, \mathbf{H}^{-1}\, \nabla_\theta L(z_m,\theta^\star). \tag{6}$$

This provides a proxy for full retraining using only gradients at $\theta^\star$ and inverse Hessian–vector products.

3.2 Three approximation layers of K-FAC for influence estimation

To make influence computation tractable at scale, K-FAC [13] and EK-FAC [14] are the key structured methods we use to approximate the Hessian that appears in Equation 6. This section states the equations we evaluate and decomposes the approximation into three layers: (i) Implicit model linearisation, (ii) block-diagonal approximation, and (iii) Pre-post activation approximation (with and without eigenvalue correction).

3.2.1 Implicit model linearisation

The first step uses the Generalised Gauss–Newton (GGN) matrix [11] as a positive-semidefinite curvature proxy for the full Hessian that removes (linearises) network curvature and only focuses on output-space curvature. The substitution avoids unstable second-derivative terms and aims to keep inversion operations well-conditioned.

Formulation. Let $u_i(\theta) = f(x_i;\theta) \in \mathbb{R}^{d_y}$, $\mathbf{J}_i(\theta) = \nabla_\theta u_i(\theta)$, $g_i(\theta) = \nabla_u \phi(u_i(\theta), y_i)$, and $\mathbf{H}_i^{(u)}(\theta) = \nabla_u^2 \phi(u_i(\theta), y_i)$. The empirical Hessian admits

$$\mathbf{H}(\theta) = \frac{1}{N}\sum_{i=1}^N \mathbf{J}_i^\top \mathbf{H}_i^{(u)} \mathbf{J}_i \;+\; \frac{1}{N}\sum_{i=1}^N \sum_{k=1}^{d_y} [g_i]_k\, \nabla_\theta^2 u_{i,k}(\theta) \;=\; \mathbf{G}(\theta) \;+\; \mathbf{R}(\theta),$$

where $\mathbf{G}$ is the GGN term and $\mathbf{R}$ is the residual collecting second-order parameter non-linearities. For exponential-family likelihoods, $\mathbf{G}$ coincides with the Fisher information matrix [11, 27].

Near-optimal convergence. First, when parameters θ yield near-optimal predictions for all training examples, the gradient of the loss with respect to outputs vanishes: $g_i(\theta) \approx 0$ for all i. In such cases, the residual $\mathbf{R}(\theta)$ becomes negligible irrespective of the model's intrinsic curvature, yielding $\mathbf{H}(\theta) \approx \mathbf{G}(\theta)$. This condition arises specifically even at local minima where $\nabla_\theta \mathcal{J}(\theta) = 0$.

Piecewise-linear activations. Second, for networks with piecewise-linear activation functions (e.g., ReLU), the output-curvature term $\mathbf{R}(\theta)$ is *block-hollow* [28–30]: within-block second derivatives (e.g., within a layer) vanish, while mixed cross-block derivatives can be non-zero [1]. Consequently, the block-diagonal of $\mathbf{H}(\theta)$ equals the block-diagonal of $\mathbf{G}(\theta)$, while the full matrices need not be equal.

Neural Tangent Kernel regime. Third, under the Neural Tangent Kernel (NTK) regime [31], where network widths are large relative to data complexity, the model output $u_i(\theta)$ remains closely approximated by its first-order Taylor expansion around initial parameters θ_0 throughout optimisation [32]. This local linearity implies $\nabla^2_\theta u_{i,k}(\theta) \approx \mathbf{0}$ along the optimisation trajectory, rendering $\mathbf{R}(\theta)$ negligible and ensuring $\mathbf{H}(\theta) \approx \mathbf{G}(\theta)$ during training. This is subject to some assumptions in initialisation of parameters and a sufficiently small learning rate.

Remark. One important remark is that the residual $\mathbf{R}(\theta)$ is not necessarily positive semi-definite and may contribute both positive and negative curvature to $\mathbf{H}(\theta)$ [11]. Discarding $\mathbf{R}(\theta)$ thus removes potentially useful curvature information beyond merely suppressing instability. The optimisation literature often prioritises $\mathbf{G}(\theta)$ due to its numerical stability and since the curvature coming from output curvature $\mathbf{H_i}^{(u)}(\theta)$ is more important than those coming from $\nabla^2_\theta u_{i,k}$ in $\mathbf{R}(\theta)$ over the training trajectory. This preference does not imply that $\mathbf{R}(\theta)$ is universally irrelevant in contexts requiring full Hessian fidelity, such as in influence functions.

3.2.2 Block-diagonal approximation

This second approximation step masks cross-layer curvature and focuses on layer-wise (or group-wise) blocks so that inversion decouples across blocks, encouraging parallelism and memory efficiency.

Formulation. Partition parameters as $\theta = (\theta_1, \dots, \theta_L)$ and approximate

$$\mathbf{G}(\theta) \;\approx\; \mathrm{diag}(\mathbf{G}_1, \dots, \mathbf{G}_L), \qquad \mathbf{G}(\theta)^{-1} \;\approx\; \mathrm{diag}(\mathbf{G}_1^{-1}, \dots, \mathbf{G}_L^{-1}).$$

Computational advantages. Block-diagonal curvature is widely used [13, 33] because the inverse of a block-diagonal matrix decomposes into the inverses of its blocks: if $\mathbf{G} = \mathrm{diag}(\mathbf{G}_1, \dots, \mathbf{G}_L)$ then $\mathbf{G}^{-1} = \mathrm{diag}(\mathbf{G}_1^{-1}, \dots, \mathbf{G}_L^{-1})$. This structural property enables parallel computation of each block's inverse, dramatically reducing computational cost from $\mathcal{O}(D^3)$ to $\mathcal{O}(\sum_i d_i^3)$ where d_i is the dimension of block i. In the optimisation literature, studies of block-diagonal methods demonstrate that simply ignoring off-block cross-terms can even yield superior convergence and generalisation compared to both full GGN and first-order optimisers, while requiring substantially less memory than full-matrix approaches [34].

Cross-layer coupling interpretation. To make precise what is discarded when one retains only the diagonal blocks, partition parameters by groups (e.g., layers) $\theta = (\theta_1, \dots, \theta_L)$ and write the output Jacobian as $\mathbf{J} = [\,\mathbf{J}_1 \;\cdots\; \mathbf{J}_L\,]$ with $\mathbf{J}_i := \partial u / \partial \theta_i \in \mathbb{R}^{d_y \times n_i}$. For a convex-in-output loss with per-example output Hessian $\mathbf{H}^{(u)}$, the GGN is $\mathbf{G} = \frac{1}{N} \sum_{i=1}^N \mathbf{J}^\top \mathbf{H}^{(u)} \mathbf{J}$, which decomposes into a block matrix $\mathbf{G} = \big[\mathbf{G}_{ij}\big]_{i,j=1}^L$ with cross–block couplings

$$\mathbf{G}_{ij} \;=\; \frac{1}{N} \sum_{k=1}^N \mathbf{J}_i(x_k)^\top \, \mathbf{H}_k^{(u)} \, \mathbf{J}_j(x_k) \quad (i \neq j).$$

These off–diagonal terms quantify, in an $\mathbf{H}^{(u)}$–weighted inner product, how similarly two parameter blocks move the model's outputs: under squared loss, $\mathbf{H}^{(u)} = \mathbf{I}$ and $\mathbf{G}_{ij}$ reduces to the Gram overlap

[1]Consider a two-layer ReLU network $u = w_2 \mathrm{ReLU}(w_1 x)$, $\partial^2 u / \partial w_1^2 = \partial^2 u / \partial w_2^2 = 0.$, but $\partial^2 u / \partial w_1 \partial w_2 = \mathrm{ReLU}'(w_1 x)\, x \neq 0$. This illustrates the block-hollow $\mathbf{R}(\theta)$.

$N^{-1}\sum_k \mathbf{J}_i(x_k)^\top \mathbf{J}_j(x_k)$, so cross–block magnitude is driven by the alignment of the two blocks' output–sensitivities.

Justification for block-diagonality. Classical analyses of one–hidden–layer MLPs reuse the same result to justify the block-diagonal structure of Hessian: [35] derives explicit off–diagonal formulas and shows that with a cross–entropy (CE) loss the factors $P_\theta(y|x)\big(1-P_\theta(y|x)\big)$ multiply those couplings, pushing them toward zero during training and yielding an (approximately) block–diagonal Hessian across units, and by contrast, with mean–squared error (MSE) the same cancellation need not occur, so off–diagonals generally persist [36]. More recently, a finite–sample–to–asymptotic theory at random initialisation proves that in linear models and in one–hidden–layer networks (under both MSE and CE) the *ratio* of off–diagonal to diagonal block norms vanishes as the number of outputs/classes C grows (with rates depending on the block), providing a justification for block–diagonal curvature when C is large, as in modern LLMs [37].

Remark. These findings, however, are based solely on experiments with one-hidden-layer MLPs, their extension to deeper architectures requires further investigation. For deeper networks the block–diagonal assumption remains an approximation whose accuracy depends on how orthogonal (in the $\mathbf{H}^{(u)}$–metric) the per–block output Jacobians become in practice, a question we will probe empirically in the next chapter, particularly in the context of influence functions where curvature information might be important.

3.2.3 Pre-post activation approximation

The last approximation, K-FAC, uses a separable Kronecker structure to reduce matrix size and enable fast inversion for each curvature block.

Formulation. For layer ℓ with bias-augmented inputs $\bar{\mathbf{a}}_{\ell-1} \in \mathbb{R}^{M+1}$ and pre-activation gradients $\mathrm{D}\mathbf{s}_\ell \in \mathbb{R}^P$, K-FAC assumes $\bar{\mathbf{a}}_{\ell-1}$ and $\mathrm{D}\mathbf{s}_\ell$ are independent and approximates the GGN/Fisher block as

$$\mathbf{G}_\ell \;\approx\; \mathbf{A}_{\ell-1} \otimes \mathbf{S}_\ell, \qquad \mathbf{A}_{\ell-1} := \mathbb{E}[\bar{\mathbf{a}}_{\ell-1}\bar{\mathbf{a}}_{\ell-1}^\top], \quad \mathbf{S}_\ell := \mathbb{E}[\mathrm{D}\mathbf{s}_\ell \mathrm{D}\mathbf{s}_\ell^\top],$$

with $(\mathbf{A}_{\ell-1} \otimes \mathbf{S}_\ell)^{-1} = \mathbf{A}_{\ell-1}^{-1} \otimes \mathbf{S}_\ell^{-1}$.

Remark. By assuming independence between $\bar{\mathbf{a}}_{\ell-1}$ and $\mathrm{D}\mathbf{s}_\ell$, K-FAC loses the cross-covariance structure that couples activations and gradients on individual examples. As a result, it cannot represent effects such as parameters that rarely activate also rarely receiving large gradients, or input patterns that jointly induce high activations and large error signals. This missing information can be substantial in non-linear networks where the coupling between $\bar{\mathbf{a}}_{\ell-1}$ and $\mathrm{D}\mathbf{s}_\ell$ helps characterise local geometry. A spectral view makes the same point: the exact GGN block admits

$$\mathbf{G}_\ell = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\top, \tag{7}$$

whereas K-FAC uses the Kronecker-factor surrogate

$$\mathbf{A}_{\ell-1} \otimes \mathbf{S}_\ell = (\mathbf{U}_A \mathbf{\Lambda}_A \mathbf{U}_A^\top) \otimes (\mathbf{U}_S \mathbf{\Lambda}_S \mathbf{U}_S^\top) = (\mathbf{U}_A \otimes \mathbf{U}_S)\,(\mathbf{\Lambda}_A \otimes \mathbf{\Lambda}_S)\,(\mathbf{U}_A \otimes \mathbf{U}_S)^\top. \tag{8}$$

Although the update rotates into the Kronecker eigenbasis $\mathbf{U}_A \otimes \mathbf{U}_S$, its rescaling uses only products of marginal spectra $\mathbf{\Lambda}_A \otimes \mathbf{\Lambda}_S$. These products $\lambda_i^A \lambda_j^S$ are generally not the true variances of the full block along $(\mathbf{U}_A \otimes \mathbf{U}_S)$'s directions, leading to systematic curvature misestimation: marginal second moments are preserved, but cross-covariances are discarded, which distorts the spectrum of the true GGN block.

3.2.4 Eigenvalue correction

Instead, we can keep K-FAC's Kronecker-factored eigenbasis while correcting the per-direction scaling to better match the empirical curvature.

Formulation. Write $\mathbf{A}_{\ell-1} = \mathbf{U}_A \mathbf{\Lambda}_A \mathbf{U}_A^\top$ and $\mathbf{S}_\ell = \mathbf{U}_S \mathbf{\Lambda}_S \mathbf{U}_S^\top$, and let $\mathbf{U} := \mathbf{U}_A \otimes \mathbf{U}_S$. If g_ℓ denotes the (vectorised) per-layer gradient, EK-FAC sets

$$s_k^\star \;:=\; \mathbb{E}\Big[\big(\mathbf{U}^\top g_\ell\big)_k^2\Big], \qquad \mathbf{S}^\star := \mathrm{diag}(s_1^\star, \ldots, s_K^\star), \qquad \mathbf{G}_\ell \;\approx\; \mathbf{U}\,\mathbf{S}^\star\,\mathbf{U}^\top.$$

Remark. However, EK-FAC does not correct the direction of the approximation: it retains K-FAC's Kronecker-factored eigenbasis $\mathbf{U}_A \otimes \mathbf{U}_S$. The update only corrects per-coordinate scaling by matching the Fisher's diagonal in that basis, so any genuine coupling between coordinates, i.e., curvature that appears as off-diagonal mass in Kronecker eigenbasis coordinates remains unmodelled. Thus, when the true block $\mathbf{G}_\ell$ has principal directions that are not well captured by a separable Kronecker structure, eigenvalue correction alone cannot recover those interactions as it rescales coordinates rather than also rotating them.

4 Investigating the approximation error & influence score relationship

4.1 Experimental setup

Objective. To understand how each approximation layer impacts influence quality, we need experimental settings where the approximation errors vary systematically. In Section 3.2 we identified that each approximation layer—GGN substitution, block-diagonalisation, and Kronecker factorisation—introduces different error types that depend on the model's curvature properties. We therefore design experiments along three dimensions that naturally modulate these curvature characteristics: (i) training duration, where early training exhibits large residual terms that diminish near convergence; (ii) network depth, which amplifies cross-layer coupling and non-linear interactions between parameters; and (iii) network width, which affects the conditioning and spectral properties of individual layer blocks. These controlled variations allow us to isolate when each approximation breaks down and quantify its impact on attribution fidelity.

Dataset. We use the Digits dataset [38], which contains $n = 1{,}797$ greyscale images of handwritten digits (0–9). Each 8×8 image is converted into a 64-dimensional vector. We randomly split the data into $n_{\text{train}} = 1{,}617$ training samples (90%) and $n_{\text{test}} = 179$ test samples, maintaining equal representation of all digit classes.

Model architecture. Due to computational constraints, we restrict our experiments to multi-layer perceptrons (MLPs). The specific training settings are specified in the hyperparameter settings section below, and the limitations of this choice are discussed in the Section 5. We use Tanh activation functions throughout, which ensure non-convexity and that the residual term exists, allowing us to isolate the effects of training.

Matrix inversion and numerical stability. As shown in Section 3.2, the Hessian is often ill-conditioned even for simple models. To address invertibility, one option is to add a Tikhonov damping term λ to the diagonal of each matrix. For this experiment, this approach biases curvature differences between methods, which makes the comparison unfair. We therefore adopt a second approach: pseudo-inverse computation. The conventional choice is the Moore–Penrose pseudo-inverse via SVD [39]. However, we instead use an eigendecomposition-based pseudo-inverse [40] for two reasons: (i) the decomposition exists for square symmetric matrices, which is the case for the Hessian; (ii) we often want to regularise the matrix to be positive definite, which is simpler with eigenvalue adjustments than with SVD.

Formally, for a symmetric matrix $\mathbf{H} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, the pseudo-inverse is $\mathbf{H}^\dagger = \mathbf{Q}\mathbf{\Lambda}^\dagger\mathbf{Q}^\top$ with diagonal entries $[\mathbf{\Lambda}^\dagger]_{ii} = \lambda_i^{-1}\,\mathbf{1}\{|\lambda_i| > \epsilon\}$ and zero otherwise. We set $\epsilon = 10^{-4}$ and use no damping in this section. Sensitivity to ϵ is potential future work and may be relevant for interpreting the results; see Section 5.

Evaluation metrics. We employ two complementary metrics to assess both the quality of influence attributions and the fidelity of Hessian approximations:

- **Linear data-modelling score (LDS):** Following the framework described in Appendix A, we use the expected leave-some-out evaluation with subset fraction α.
- **Approximation error:** We cannot reliably use $\mathbf{H}^{-1}$ as a reference because $\mathbf{H}$ is typically singular or nearly singular in our setting; instead we assess whether $\mathbf{H}\widehat{\mathbf{H}}^{-1}v \approx v$. For a set of vectors

$\{v_i\}_{i=1}^{N}$ (using training data gradients), we compute:

$$\text{Approximation Error} = \frac{1}{N}\sum_{i=1}^{N}\frac{\|\mathbf{H}\cdot\widehat{\mathbf{H}}^{-1}v_i - v_i\|^2}{\|v_i\|^2} \tag{9}$$

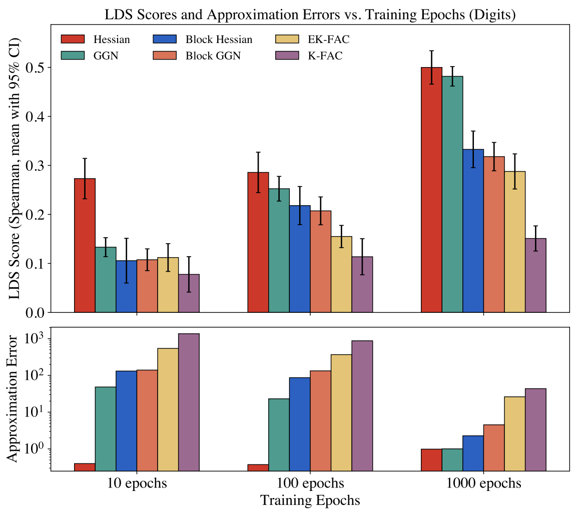
4.2 Results

We now present our empirical findings addressing the three core questions posed in the introduction, examining the relationship between approximation fidelity and attribution quality across our controlled experimental conditions.

(1) Does a better Hessian approximation improve influence scores?

Across all settings we find a consistent inverse relationship between curvature approximation error and influence fidelity: lower error corresponds to higher LDS, with the method ordering Hessian $\gtrsim$ GGN $>$ Block-GGN $>$ EK-FAC $>$ K-FAC visible in the top (LDS) and bottom (error) panels of Figure 1–3. The slope of this association depends on training stage and architecture. Along training, moving from 10 to 100 to 1000 epochs tightens the cloud of method points in Figure 1: approximation error decreases while LDS increases and then saturates, and the methods cluster near convergence, indicating diminishing marginal LDS gains from additional curvature fidelity late in training. With architecture, increasing depth lowers LDS and raises approximation error for all methods (Figure 2), whereas width produces smaller movements with the same ordering (Figure 3).

Two diagnostics account for the stage- and architecture-dependence: cross-layer coupling (off-block mass) decreases mildly over training and increases strongly with depth (Appendix Figure 6), so the LDS–error slope is flatter at late epochs (weaker cross-block terms) and steeper in deeper networks (stronger cross-block terms). In addition, Kronecker spectral fidelity improves with training and worsens with depth, with EK-FAC showing consistently higher eigenvalue overlap than K-FAC (Appendix Figure 7) while the two share the same Kronecker eigenbasis and both exhibit declining basis alignment with depth (Appendix Figure 8). These properties explain why EK-FAC sits consistently above K-FAC in LDS yet remains below unfactorised Block-GGN, and why depth amplifies between-method gaps whereas width does not.



Step (incremental)	ΔES%	ΔLDS%
Epoch = 10		
Hessian → GGN	3.52	**-71.59**
GGN → B-GGN	6.70	-13.12
B-GGN → EK-FAC	29.58	2.29
EK-FAC → K-FAC	**60.20**	-17.58
Epoch = 100		
Hessian → GGN	2.59	-19.35
GGN → B-GGN	12.55	-26.25
B-GGN → EK-FAC	26.80	**-30.39**
EK-FAC → K-FAC	**58.06**	-24.01
Epoch = 1,000		
Hessian → GGN	0.04	-5.19
GGN → B-GGN	8.25	**-46.95**
B-GGN → EK-FAC	**50.74**	-8.67
EK-FAC → K-FAC	40.97	-39.19

Figure 1: ***Left:*** **Attribution quality vs. Hessian approximation error - Training duration.** LDS and approximation error (Equation 9); for epoch {10, 100, 1,000}. Setting is fixed at depth = 8 and width = 16; other hyperparameters follow Table 1. ***Right:*** **Error decomposition table**: incremental shares along the curvature-approximation path. ΔES% denotes Error Share in percentage in the Hessian→K-FAC path and ΔLDS% denotes the total Hessian→K-FAC LDS percentage change across steps. B-GGN denotes Block-Diagonal GGN.

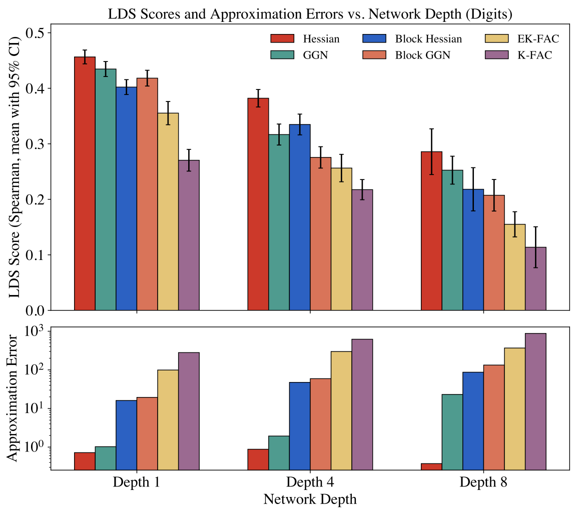
(2) Which approximation layer contributes most to the error, and what caused it?

The dominant contributor to the total Hessian→K-FAC error gap is the within-block Kronecker factorisation. In the epoch sweep (Figure 1, right tables), the incremental EK-FAC→K-FAC step accounts for $\sim$60.2% of the gap at 10 epochs, $\sim$58.1% at 100 epochs, and $\sim$41.0% at 1000 epochs. Across depth (Figure 2, right), the same step remains the largest single share ($\sim$64.8% at depth 1, $\sim$ 52.0% at depth 4, $\sim$ 58.1% at depth 8). For width (Figure 3, right), a local exception occurs at 64 units where the Block-GGN→EK-FAC share ($\sim$ 39.2%) slightly exceeds EK-FAC→K-FAC ($\sim$ 35.4%), but taken together the two factorisation steps explain the majority of the gap at every width (about 70–78%).

Further diagonstic plots also clarify the mechanism. EK-FAC and K-FAC operate in the same Kronecker eigenbasis (identical basis-overlap curves; Appendix Figure 8), so moving from EK-FAC to K-FAC primarily introduces *spectral mis-scaling* rather than basis error; correspondingly EK-FAC achieves higher eigenvalue overlap than K-FAC (Appendix Figure 7) but cannot close the gap to unfactorised blocks because the basis itself diverges from the true block basis as depth grows (Appendix Figure 8). The block-diagonal step (GGN→Block-GGN) contributes a smaller but increasing share with depth (Figure 2, right), consistent with the rise of cross-layer mass in Appendix Figure 6. By contrast, the GGN substitution (Hessian→GGN) contributes little to the total error budget except early in training (Figure 1, right), which aligns with the visual compression of method differences near convergence in Figure 1.

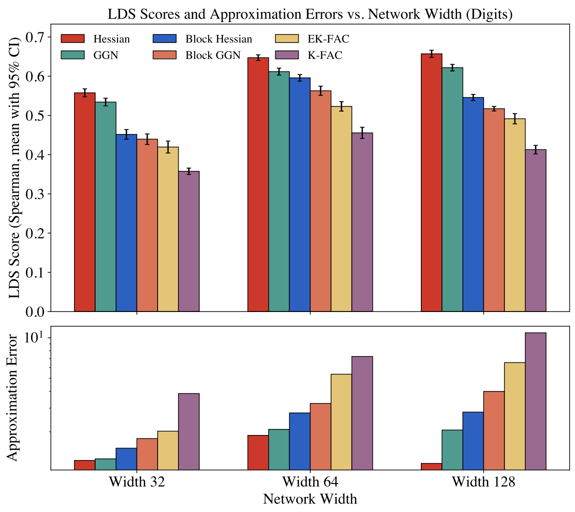
(3) Which approximation error is influence fidelity most sensitive to?

Sensitivity of the relationship between approximation error and influence fidelity is also stage- and architecture-dependent. Early in training, influence fidelity is most sensitive to the Hessian→GGN substitution: at 10 epochs a small error share ($\approx$3.5%) coincides with a large LDS drop (≈ -71.6 pp; Figure 1, right), whereas by 100 and 1000 epochs both the share and the LDS impact are much smaller (Figure 1). With increasing depth, sensitivity shifts toward block-diagonality: removing cross-block terms yields larger LDS losses per unit of error as off-block mass increases (compare, e.g., GGN→Block-GGN at depth 1 vs. 8 in Figure 2, right; see also Appendix Figure 6). Within blocks, factorisation produces the largest absolute error shares but only moderate per-share LDS penalties: EK-FAC's spectral correction improves LDS relative to K-FAC (Figure 1–3), yet both share the same eigenbasis and therefore cannot recover LDS lost to basis mismatch when depth is large (Appendix 8, with the associated eigenvalue trends in Appendix 7). Width manipulations induce comparatively small and smooth changes; at width 64 the Block-GGN→EK-FAC share slightly exceeds EK-FAC→K-FAC (3, right), but this does not alter the qualitative ordering.



Step (incremental)	ΔES%	ΔLDS%
Depth = 1		
Hessian → GGN	0.11	-11.70
GGN → B-GGN	6.57	-8.84
B-GGN → EK-FAC	28.53	-33.83
EK-FAC → K-FAC	**64.79**	**-45.63**
Depth = 4		
Hessian → GGN	0.17	**-39.73**
GGN → B-GGN	9.23	-25.00
B-GGN → EK-FAC	38.59	-11.67
EK-FAC → K-FAC	**52.01**	-23.60
Depth = 8		
Hessian → GGN	2.59	-19.35
GGN → B-GGN	12.55	-26.25
B-GGN → EK-FAC	26.80	**-30.39**
EK-FAC → K-FAC	**58.06**	-24.01

Figure 2: ***Left:*** **Attribution quality vs. Hessian approximation error - Network depth.** LDS and approximation error (Equation 9); for depth $\{1, 4, 8\}$. Setting is fixed at epoch $= 100$ and width $= 16$; other hyperparameters follow Table 1. ***Right:*** **Error decomposition table**: incremental shares along the curvature-approximation path. ΔES% denotes Error Share in percentage in the Hessian→K-FAC path and ΔLDS% denotes the total Hessian→K-FAC LDS percentage change across steps. B-GGN denotes Block-Diagonal GGN.



Step (incremental)	ΔES%	ΔLDS%
Width = 32		
Hessian → GGN	1.32	11.77
GGN → B-GGN	19.88	**47.29**
B-GGN → EK-FAC	9.11	10.03
EK-FAC → K-FAC	**69.69**	30.91
Width = 64		
Hessian → GGN	3.79	18.57
GGN → B-GGN	21.61	25.41
B-GGN → EK-FAC	**39.23**	20.80
EK-FAC → K-FAC	35.37	**35.22**
Width = 128		
Hessian → GGN	9.23	14.47
GGN → B-GGN	19.85	**42.82**
B-GGN → EK-FAC	26.17	10.48
EK-FAC → K-FAC	**44.75**	32.23

Figure 3: ***Left:* Attribution quality vs. Hessian approximation error - Network width.** LDS and approximation error (Equation 9); for widths $\{32, 64, 128\}$. Setting is fixed at epoch $= 100$ and depth $= 1$; other hyperparameters follow Table 1. ***Right:* Error decomposition table**: incremental shares along the curvature-approximation path. ΔES% denotes Error Share in percentage in the Hessian→K-FAC path and ΔLDS% denotes the total Hessian→K-FAC LDS percentage change across steps. B-GGN denotes Block-Diagonal GGN.

5 Limitations

Architecture and scale. The study uses a small MLP to keep the ELSO/LDS protocol tractable. This restricts depth, width, and dataset size, and narrows the curvature regimes observed. Results on absolute LDS levels and method ordering may not transfer to larger models. Very wide networks can operate closer to NTK-like regimes where the residual $\mathbf{R}(\theta)$ between the Hessian and the GGN is smaller, potentially changing the relative benefits of linearisation, block-diagonalisation, and factorisation. In addition, the evaluation excludes CNNs and transformers, whose curvature structure differs due to weight sharing, attention, and embeddings. Replication at larger scales and on these architectures is required before drawing general conclusions.

Evaluation design and compute budget. For Digits we use $\alpha = 0.5$, $K = 100$ groups, and $R = 50$ seeds (about $5{,}000$ retrainings per setting; Table 1). This budget limits hyperparameter sweeps, the number of datasets, and repeated width–depth grids. Although ELSO reduces variance relative to leave-one-out, credible intervals remain non-negligible in early-epoch and deep settings. Larger-scale repetitions would improve precision.

Numerical controls and regularisation comparability. In Section 4.2 we compute inverse–Hessian–vector products using an eigendecomposition pseudo-inverse with a hard threshold $\varepsilon = 10^{-4}$ and no damping ($\lambda = 0$; Equation 4.1). Truncation stabilises solves but introduces bias by discarding small-magnitude modes. We do not ablate ε, nor compare against pure Tikhonov damping $(\mathbf{G} + \lambda\mathbf{I})^{-1}$ without truncation, so sensitivity to these controls is unknown. A systematic ablation should sweep $\varepsilon \in [10^{-8}, 10^{-2}]$ and $\lambda \in [10^{-8}, 10^{-1}]$ on logarithmic grids, report LDS, and log solver pathologies (non-convergence, extreme IHVP norms). Two edge cases are also informative: using only damping with no truncation, and using no damping with an extremely small truncation threshold, to separate numerical effects from approximation quality.

6 Conclusion

Our study provides an empirical answer to the three questions posed in the introduction. We decomposed common approximations into implicit linearisation, block-diagonal structure, and Kronecker factorisation, and evaluated attribution fidelity under expected leave-some-out retraining. Across training stages and architectural choices, better curvature fidelity generally aligned with stronger attribution, while gains narrowed near convergence. The dominant source of degradation arose from Kronecker factorisation within blocks; eigenvalue-corrected variants reduced but did not remove this gap.

Acknowledgements

We thank Felix Dangel for pointing out a mistake in an earlier version of this work.

Runa Eschenhagen is supported by ARM, the Cambridge Trust, and the Qualcomm Innovation Fellowship. Bruno Mlodozeniec is supported by the Qualcomm Innovation Fellowship. Richard E. Turner is supported by the EPSRC Probabilistic AI Hub (EP/Y028783/1).

References

[1] Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, 2024.

[2] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017. URL https://arxiv.org/abs/1703.04730.

[3] 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.

[4] Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. Influence selection for active learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9274–9283, 2021.

[5] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019.

[6] Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

[7] Bruno Mlodozeniec, Runa Eschenhagen, Juhan Bae, Alexander Immer, David Krueger, and Richard Turner. Influence functions for scalable data attribution in diffusion models. *arXiv preprint arXiv:2410.13850*, 2024. URL https://arxiv.org/abs/2410.13850.

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

[9] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain. ., 1994.

[10] Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40, 2017.

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

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

[13] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2408–2417. PMLR, 2015.

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

[15] Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In *International conference on machine learning*, pages 1817–1853. PMLR, 2022.

[16] Thomas T Zhang, Behrad Moniri, Ansh Nagwekar, Faraz Rahman, Anton Xue, Hamed Hassani, and Nikolai Matni. On the concurrence of layer-wise preconditioning methods and provable feature learning. *arXiv preprint arXiv:2502.01763*, 2025.

[17] Felix Dangel, Lukas Tatzel, and Philipp Hennig. Vivit: Curvature access through the generalized gauss-newton's low-rank structure. *arXiv preprint arXiv:2106.02624*, 2021.

[18] Runa Eschenhagen, Alexander Immer, Richard Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. *Advances in Neural Information Processing Systems*, 36:33624–33655, 2023.

[19] Nikolaos Tselepidis, Jonas Kohler, and Antonio Orvieto. Two-level k-fac preconditioning for deep learning. *arXiv preprint arXiv:2011.00573*, 2020.

[20] Andrew Wang, Elisa Nguyen, Runshi Yang, Juhan Bae, Sheila A McIlraith, and Roger Grosse. Better training data attribution via better inverse hessian-vector products. *arXiv preprint arXiv:2507.14740*, 2025.

[21] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

[22] Jacob R Epifano, Ravi P Ramachandran, Aaron J Masino, and Ghulam Rasool. Revisiting the fragility of influence functions. *Neural Networks*, 162:581–588, 2023.

[23] Bruno Mlodozeniec, Isaac Reid, Sam Power, David Krueger, Murat Erdogdu, Richard E Turner, and Roger Grosse. Distributional training data attribution. *arXiv preprint arXiv:2506.12965*, 2025.

[24] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32, 2019.

[25] Xichen Ye, Yifan Wu, Weizhong Zhang, Cheng Jin, and Yifan Chen. Towards robust influence functions with flat validation minima. *arXiv preprint arXiv:2505.19097*, 2025.

[26] Hongbo Zhu and Angelo Cangelosi. Revisiting data attribution for influence functions. *arXiv preprint arXiv:2508.07297*, 2025.

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

[28] Sidak Pal Singh, Gregor Bachmann, and Thomas Hofmann. Analytic insights into structure and rank of neural network hessian maps. *Advances in Neural Information Processing Systems*, 34: 23914–23927, 2021.

[29] Sidak Pal Singh, Thomas Hofmann, and Bernhard Schölkopf. The hessian perspective into the nature of convolutional neural networks. *arXiv preprint arXiv:2305.09088*, 2023.

[30] Felix Dangel, Stefan Harmeling, and Philipp Hennig. Modular block-diagonal curvature approximations for feedforward architectures. In *International Conference on Artificial Intelligence and Statistics*, pages 799–808. PMLR, 2020.

[31] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[32] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

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

[34] Huishuai Zhang, Caiming Xiong, James Bradbury, and Richard Socher. Block-diagonal hessian-free optimization for training neural networks. *arXiv preprint arXiv:1712.07296*, 2017.

[35] Ronan Collobert. Large scale machine learning. *Idiap Res. Inst., Martigny, Switzerland, RR-04-42*, 2004.

[36] Zhaorui Dong, Yushun Zhang, Zhi-Quan Luo, Jianfeng Yao, and Ruoyu Sun. Towards quantifying the hessian structure of neural networks. *arXiv preprint arXiv:2505.02809*, 2025.

[37] Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. *Advances in Neural Information Processing Systems*, 37:114147–114179, 2024.

[38] Ethem Alpaydin and Cenk Kaynak. Optical Recognition of Handwritten Digits [Dataset], 1998. URL https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits.

[39] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[40] Stephen Wright, Jorge Nocedal, et al. Numerical optimization. *Springer Science*, 35(67-68):7, 1999.

[41] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023. URL https://arxiv.org/abs/2303.14186.

[42] Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentiation. *arXiv preprint arXiv:2405.12186*, 2024. URL https://arxiv.org/abs/2405.12186.

[43] Andrew Ilyas and Logan Engstrom. Magic: Near-optimal data attribution for deep learning. *arXiv preprint arXiv:2504.16430*, 2025.

A Measuring Attribution Quality

A.1 Linear Data-modelling Score

The *Linear Data-modelling Score* (LDS) provides a metric for evaluating the fidelity of training data attribution methods by simply taking the rank correlation between the ground truth scores against the predicted attribution scores. To compute the LDS, draw m random subsets $S_1, \ldots, S_m \sim \mathcal{D}$ and for each sample/subset S_j measure the true model outcome $f(S_j)$ (e.g., loss or accuracy after training on S_j) and the influence prediction $\hat{f}(S_j)$ provided by the attribution method. The LDS is then defined as the Spearman rank correlation:

$$\mathrm{LDS}_{\mathcal{D}}(\hat{f}) \;=\; \rho\big(\{(f(S_j),\, \hat{f}(S_j))\}_{j=1}^{m}\big), \tag{10}$$

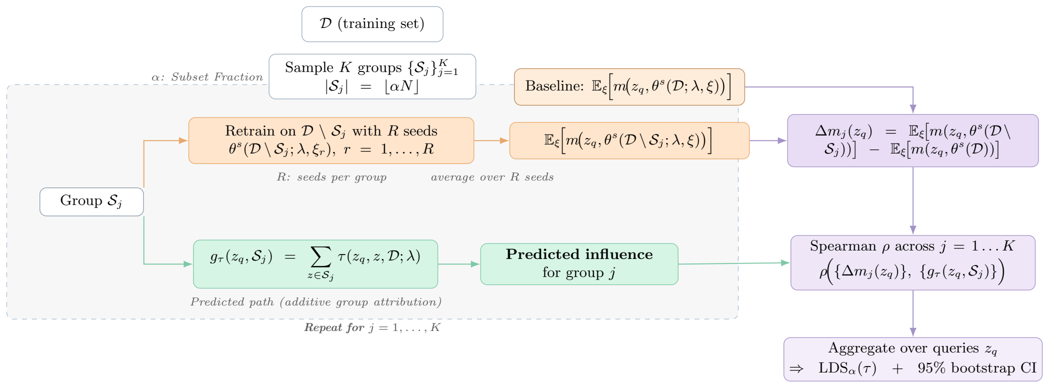
capturing how faithfully $\hat{f}$ predicts true model behaviour across subsets drawn from $\mathcal{D}$. A higher LDS thus indicates stronger predictive fidelity under the data distribution.

A.2 Expected Leave-Some-Out Retraining

The central framework being used in this work is proposed by [41] and has also been implemented in recent works [42, 20, 43]. It evaluate TDA methods by measuring their ability to predict the effect of removing *groups* of training examples rather than individual points, the ground truth thereof can be referred to as *expected leave-some-out* (ELSO) retraining. For a TDA method τ that assigns attribution scores to training examples, we leverage the additive nature of most attribution methods to compute group attributions. Given a subset $\mathcal{S} \subset \mathcal{D}$ of the training data, the group attribution for a query point z_q is:

$$g_\tau(z_q, \mathcal{S}, \mathcal{D}; \lambda) := \sum_{z \in \mathcal{S}} \tau(z_q, z, \mathcal{D}; \lambda), \tag{11}$$

where λ represents the hyperparameters used for training. For influence functions, this additivity follows naturally from the linearity of the first-order Taylor approximation.



$\mathcal{D}$ (training set)
Sample K groups $\{\mathcal{S}_j\}_{j=1}^K$ $|\mathcal{S}_j| = \lfloor \alpha N \rfloor$
α: Subset Fraction
Baseline: $\mathbb{E}_\xi\big[m(z_q, \theta^s(\mathcal{D}; \lambda, \xi))\big]$
Retrain on $\mathcal{D} \setminus \mathcal{S}_j$ with R seeds $\theta^s(\mathcal{D} \setminus \mathcal{S}_j; \lambda, \xi_r),\ r = 1, \ldots, R$
$\mathbb{E}_\xi\big[m(z_q, \theta^s(\mathcal{D} \setminus \mathcal{S}_j; \lambda, \xi))\big]$
$\Delta m_j(z_q) = \mathbb{E}_\xi[m(z_q, \theta^s(\mathcal{D} \setminus \mathcal{S}_j))] - \mathbb{E}_\xi[m(z_q, \theta^s(\mathcal{D}))]$
R: seeds per group
average over R seeds
Group $\mathcal{S}_j$
$g_\tau(z_q, \mathcal{S}_j) = \sum_{z \in \mathcal{S}_j} \tau(z_q, z, \mathcal{D}; \lambda)$
Predicted influence for group j
Spearman ρ across $j = 1 \ldots K$ $\rho\big(\{\Delta m_j(z_q)\}, \{g_\tau(z_q, \mathcal{S}_j)\}\big)$
Predicted path (additive group attribution)
Repeat for $j = 1, \ldots, K$
Aggregate over queries z_q $\Rightarrow$ $\mathrm{LDS}_\alpha(\tau)$ + 95% bootstrap CI

Figure 4: **Expected leave-some-out LDS evaluation.** We sample K random subsets of the training data, retrain the model R times per subset to average out randomness, and measure the resulting change in the query metric relative to the full-data baseline. We predict each group's effect by summing per-example attributions, then report the Spearman rank correlation between observed and predicted effects across groups, aggregated over queries with 95% bootstrap confidence intervals.

The LDS evaluation proceeds through the following systematic approach:

1. **Subset generation**: We generate K random subsets $\{\mathcal{S}_j\}_{j=1}^K$ from the training dataset, each containing $\lfloor \alpha N \rfloor$ data points, where $\alpha \in (0, 1)$ is the data sampling ratio. This sampling ratio is crucial, too small and we lack signal, too large and we approach the computational cost of full retraining.
2. **Model retraining**: For each training run with removed subset $\mathcal{S}_j$, we train the model R times with different random seeds (controlling initialisation and batch ordering):

$$\{\theta^s(\mathcal{S}_j; \lambda, \xi_r)\}_{r=1}^R, \tag{12}$$

where ξ_r represents the r-th random seed.

3. **Calculating correlation**: For a query point z_q, we compute the Spearman rank correlation between:
    - The predicted group attributions: $\{g_\tau(z_q, \mathcal{S}_j, \mathcal{D}; \lambda) : j \in [K]\}$
    - The actual measured effects: $\left\{ \frac{1}{R} \sum_{r=1}^{R} m\big(z_q, \theta^s(\mathcal{S}_j; \lambda, \xi_r)\big) : j \in [K] \right\}$
4. **Aggregation**: The final LDS is computed by averaging correlations across multiple query points:

$$\mathrm{LDS}_\alpha(z_q, \tau) = \rho\Big(\mathbb{E}_\xi\big[m(z_q, \theta^s(\mathcal{S}_j; \lambda, \xi))\big] : j \in [K],\ \{g_\tau(z_q, \mathcal{S}_j, \mathcal{D}; \lambda) : j \in [K]\}\Big), \tag{13}$$

where ρ denotes the Spearman rank correlation.

To ensure robust results, we report LDS scores with 95% bootstrap confidence intervals, accounting for the randomness in subset selection.

B Hyperparameter Settings

Table 1 summarises the training details for all experiments. We selected these hyperparameters through preliminary experiments to ensure models achieve reasonable convergence while maintaining computational tractability.

Dataset	Architecture	Training	ELSO Retrain

Digits
Train: 1,617
Query: 179
	MLP
	Depth: {1, 4, 8}
	Width: {32, 64, 128}
		SGD w/ Scheduler
		Learning rate: 0.03
		Weight decay: 0
		Batch size: 32
		Epochs: {10, 100, 1000}
			Leave-Some-Out
			α: 0.5
			R: 50
			K: 100
			Total: 5,000 Models

Table 1: **Summary of training details for Digits dataset.**

For the experiments varying model architecture, we modify either the depth (1, 4, or 8 layers) or width (32, 64, or 128 hidden units per layer) while keeping other hyperparameters fixed. For the training duration experiments, we evaluate models at 10, 100, and 1000 epochs. We employ a Cosine scheduler to smoothly anneal the learning rate over training, which promotes more stable convergence. The settings for ELSO retraining follow those of Bae et al. [42].

C Attributing the Approximation Error

We now decompose the curvature matrices into the components that most influence the approximation error quantified in the previous section. Our aim is descriptive: to identify what changes across training time, depth, and width, and to relate these changes to the approximation path introduced earlier. Throughout, $\mathbf{H}$ denotes the exact Hessian, $\mathbf{G}$ the GGN, and $\mathbf{BG}$ is the exact block-diagonal GGN. For factored methods we write $\widehat{\mathbf{G}} \in \{\text{K-FAC}, \text{EK-FAC}\}$. Figures 5–8 summarise the measurements over the three sweeps (epochs, depth, width), and we refer back to the previous subsection for the corresponding error–LDS co-movement.

C.1 Residual Curvature

In order to quantify how much of the Hessian's norm is captured by the GGN, we use the residual magnitude and track it over training epochs, network depth, and width:

$$r_{\mathrm{rel}} = \frac{\|\mathbf{H} - \mathbf{G}\|_F}{\|\mathbf{H}\|_F}. \tag{14}$$

Figure 5 shows a pronounced decline in the residual magnitude over the training duration, followed by a plateau at late epochs. This indicates that, as optimisation proceeds, the GGN accounts for an increasingly large fraction of the Hessian's norm. Across network depth, the residual ratio increases from shallow to deep models, yielding a clear monotone trend. By comparison, width manipulations induce small, non-monotone changes with a much smaller dynamic range than either training time or depth. These observations align with the linearisation perspective summarised in Section 3.2: near stationary points the Taylor remainder term that separates $\mathbf{H}$ from $\mathbf{G}$ becomes small, whereas deeper networks, by construction, sustain a larger residual even at comparable training loss. In the context of the results in Figure 1, this explains why the Hessian $\to$GGN increment contributes little to the total approximation gap at late epochs.

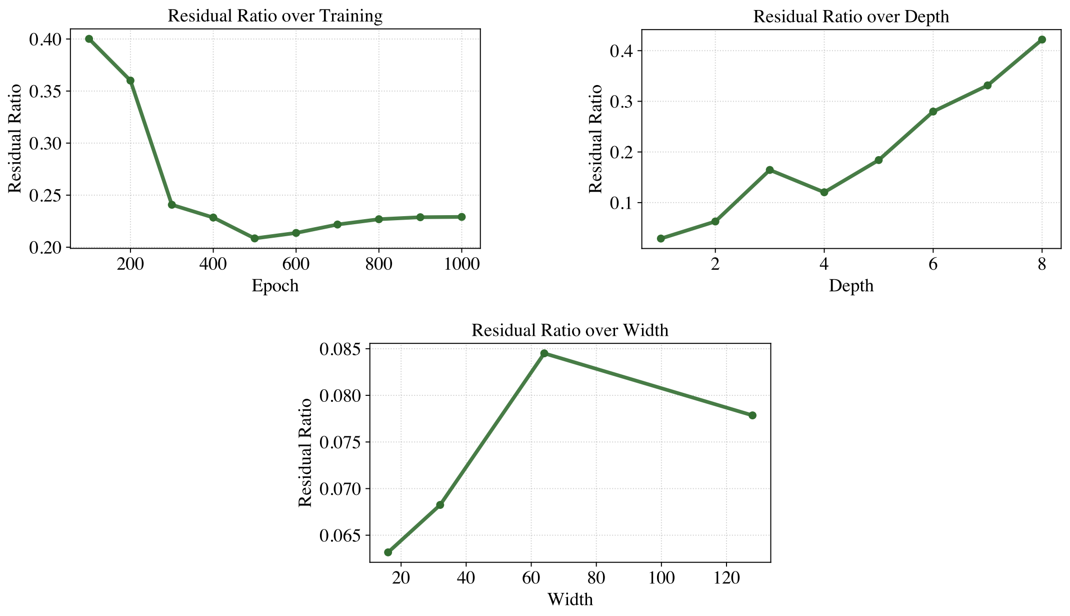


Figure 5: **Residual Term Magnitude (Digits).** Fractional size of the residual $\mathbf{R}$ relative to the Hessian $\mathbf{H}$ across (*left*) training epochs, (*right*) network depth, and (*bottom*) network width. Lower values means that $\mathbf{G}$ accounts for a larger share of $\mathbf{H}$.

C.2 Cross-Layer Curvature

In order to measure cross-layer coupling within the GGN, we use the cross-layer curvature Frobenius norm and track it over training epochs, depth, and width:

$$\rho_{\text{cross}} = \frac{\left\|\mathbf{G} - \mathbf{BG}\right\|_F}{\left\|\mathbf{G}\right\|_F}. \tag{15}$$

The cross-layer curvature in Figure 6 decreases slightly over training, indicating weaker cross-layer coupling as the model approaches its late-epoch operating point. In contrast, ρ_{off} increases strongly with depth: deeper architectures exhibit substantially larger off-block mass in the GGN. This trend stands in contrast to the common heuristic discussed in Section 3.2.2 that classification heads induce near block-diagonality; here, the measured cross-layer curvature expands with additional hidden layers. Width has a smaller and smoother effect: ρ_{off} rises gradually with width but remains well below the magnitude changes driven by depth. The stepwise results in Figure 2 are consistent with these measurements: the GGN $\to$Block-GGN increment exhibits a non-negligible ΔLDS that tracks the level of off-block mass, particularly as depth increases.

C.3 Eigen-Spectrum Alignment

In order to assess the spectral fidelity of Kronecker-factorised approximations, we use an eigenvalue-overlap metric between $\widehat{\mathbf{G}}$ and the block-diagonal GGN ($\mathbf{BG}$), tracked across epochs, depth, and

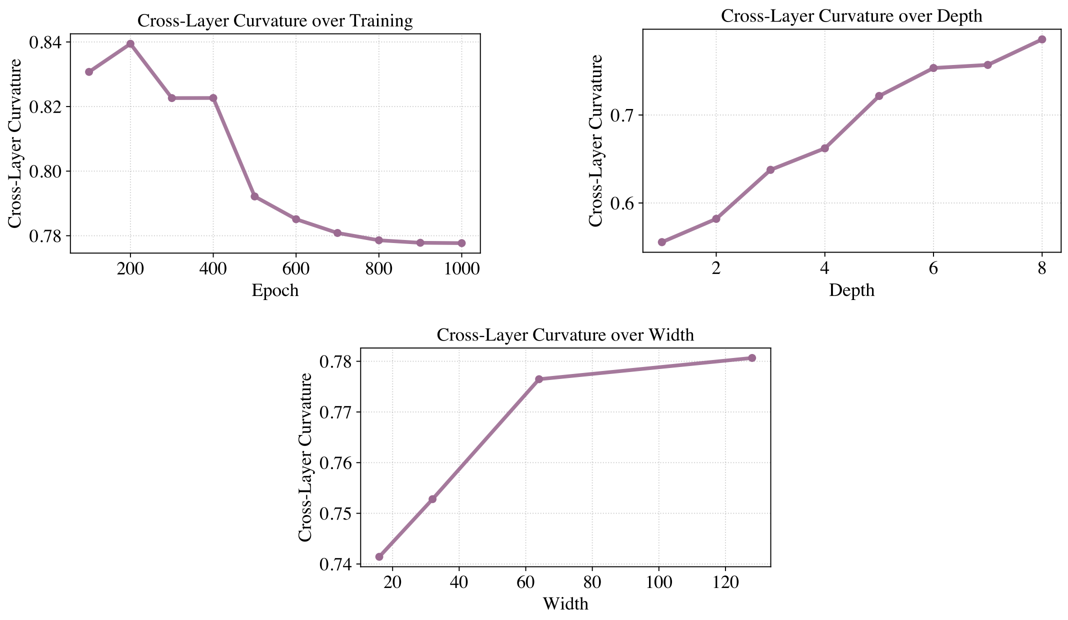


Figure 6: **Cross-layer Curvature (Digits).** Cross-layer mass of the GGN shown across (*left*) training epochs, (*right*) depth, and (*bottom*) width. Higher values indicate stronger cross-block coupling.

width:

$$\mathrm{EvalOverlap}(\widehat{\mathbf{G}}, \mathbf{BG}) \;=\; 1 \;-\; \frac{\big\|\,\mathrm{sort}\big(\lambda(\widehat{\mathbf{G}})\big) - \mathrm{sort}\big(\lambda(\mathbf{BG})\big)\big\|_2}{\big\|\,\mathrm{sort}\big(\lambda(\mathbf{BG})\big)\big\|_2}. \tag{16}$$

For combining Equation 16 across all blocks, we compute each quantity per block and aggregate via a parameter count weighted average:

$$\mathrm{Agg} \;=\; \sum_{l=1}^{L} w_l \,\mathrm{Metric}_l, \qquad w_l \;=\; \frac{d_l}{\sum_{l'=1}^{L} d_{l'}}, \tag{17}$$

with d_l the dimensionality of block L.

Figure 7 reports the aggregated overlap of eigenvalues for K-FAC and EK-FAC relative to $\mathbf{BG}$. Over training epochs, both methods improve, with EK-FAC consistently above K-FAC and reaching a higher plateau. With increasing depth, the overlap declines for both, and the gap between EK-FAC and K-FAC widens, indicating that eigenvalue misestimation becomes more severe for the stricter Kronecker factorisation as the network deepens. Changes with width are mild and positive on average. These spectral trends mirror the recovery reported in Figure 1–3: EK-FAC reduces a substantial portion of K-FAC's deficit yet does not match the full block-diagonal GGN.

C.4 Kronecker-Factored Eigenbasis Alignment

In order to evaluate Kronecker eigenbasis alignment, we use an eigenbasis-overlap metric between the eigenspaces of $\widehat{\mathbf{G}}$ and $\mathbf{BG}$, tracked across epochs, depth, and width:

$$\mathrm{BasisOverlap}(\widehat{\mathbf{G}}, \mathbf{BG}) \;=\; \frac{1}{k}\,\big\|\mathbf{V}_k(\mathbf{BG})^\top\,\mathbf{U}_k(\widehat{\mathbf{G}})\big\|_F^2 \;=\; \frac{1}{k}\sum_{i=1}^{k}\cos^2\theta_i, \tag{18}$$

where $\mathbf{V}_k(\cdot)$ and $\mathbf{U}_k(\cdot)$ collect the k eigenvectors, and $\{\theta_i\}$ are principal angles between the corresponding subspaces. Also, $\mathbf{V}_k$ and $\mathbf{U}_k$ are chosen to be the eigenvectors in the order corresponding to the eigenvalues of each block. We choose k to be the top 20% of the sorted eigenvalues, which extends prior works [17]. Similar to Equation 17, we also use parameter count weighted average across blocks to provide a combined metric.

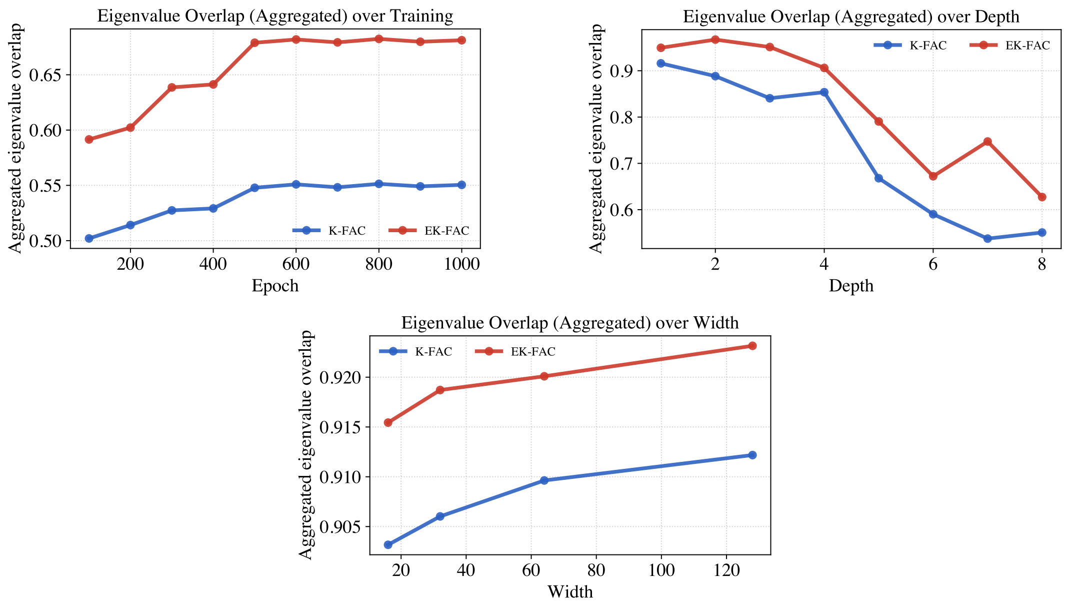


Figure 7: **Eigen-spectrum Alignment (Digits).** Aggregated eigenvalue overlap between each approximation (K-FAC, EK-FAC) and the full block GGN across (*left*) training epochs, (*right*) depth, and (*bottom*) width. Higher values indicate closer spectral agreement.

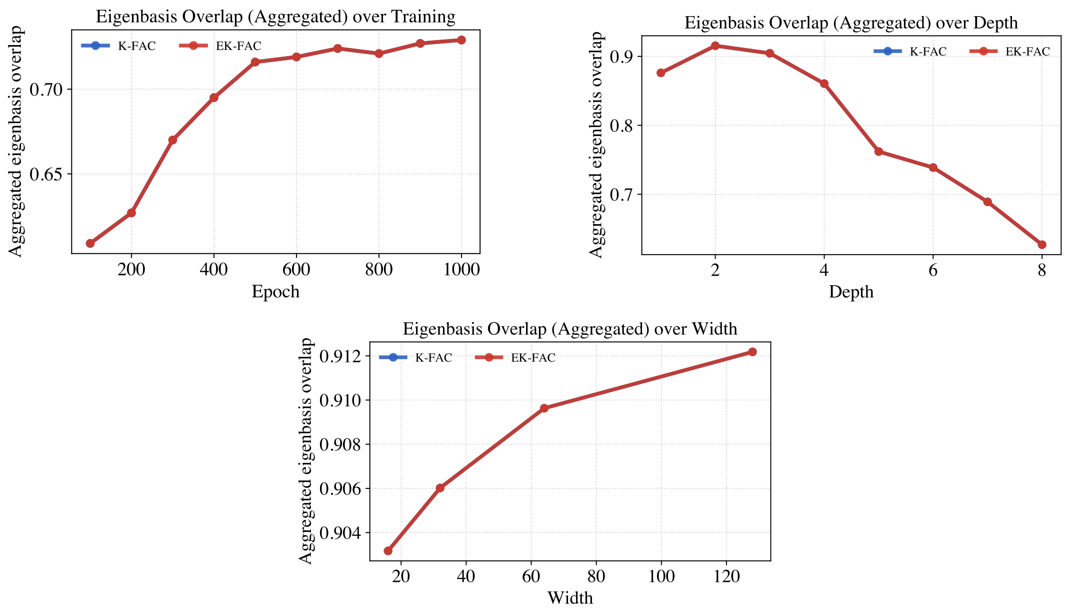


Figure 8: **Eigenbasis Alignment (Digits).** Aggregated overlap between the Kronecker-factored eigenbasis and the true eigenvectors of the full block GGN for K-FAC and EK-FAC across (*left*) training epochs, (*right*) depth, and (*bottom*) width. Higher values indicate closer basis alignment. The two methods share the same basis, which explains the perfectly aligned plots.

Figure 8 shows analogous results for subspace alignment. Over training, the basis overlap increases and the two factorisations nearly coincide at late epochs. Increasing depth produces a marked decline in overlap, with K-FAC degrading more quickly than EK-FAC; this echoes the persistent dominance of the Block-GGN $\rightarrow$K-FAC increment in the total error share. Width effects are comparatively small, with EK-FAC trending flat-to-slightly-up and K-FAC showing a shallow peak at mid width followed by a modest dip.

These plots are consistent with the fact that EK-FAC operates in the same Kronecker eigenbasis as K-FAC and changes only the per direction scaling in that basis by setting the diagonal to the second moment of the projected gradient, which is the Frobenius optimal diagonal for the chosen basis; therefore both approximations share eigenvectors and exhibit identical eigenbasis overlap with the block diagonal GGN while differing primarily in eigenvalue agreement. As training proceeds the Kronecker eigenbasis tends to decorrelate gradient coordinates relative to the parameter basis, so a diagonal model in that basis becomes effective and the diagonal correction explains the observed improvement without altering the basis itself. However, as we observe that the eigenbasis overlap with the block diagonal GGN decreases as depth increases, which indicates growing basis mismatch and persistent off diagonal mass that no method constrained to be diagonal in the Kronecker factored eigenbasis can remove; consequently the approximation quality of EK-FAC worsens with depth even though its diagonal remains optimal for that fixed basis.