Scalable Multi-Stage Influence Function for Large Language Models via Eigenvalue-Corrected Kronecker-Factored Parameterization

Yuntai Bao[1], **Xuhong Zhang**[1*], **Tianyu Du**[1*], **Xinkui Zhao**[1], **Jiang Zong**[2], **Hao Peng**[3] and **Jianwei Yin**[1]
[1]Zhejiang University
[2]Universal Identification Technology (Hangzhou) Co.,Ltd.
[3]Zhejiang Normal University
{yuntaibao, zhangxuhong, zjradty, zhaoxinkui}@zju.edu.cn,
zongj@kingflying.cn, hpeng@zjnu.edu.cn, zjuyjw@cs.zju.edu.cn

Abstract

Pre-trained large language models (LLMs) are commonly fine-tuned to adapt to downstream tasks. Since the majority of knowledge is acquired during pre-training, attributing the predictions of fine-tuned LLMs to their pre-training data may provide valuable insights. Influence functions have been proposed as a means to explain model predictions based on training data. However, existing approaches fail to compute "multi-stage" influence and lack scalability to billion-scale LLMs.

In this paper, we propose the multi-stage influence function to attribute the downstream predictions of fine-tuned LLMs to pre-training data under the full-parameter fine-tuning paradigm. To enhance the efficiency and practicality of our multi-stage influence function, we leverage Eigenvalue-corrected Kronecker-Factored (EK-FAC) parameterization for efficient approximation. Empirical results validate the superior scalability of EK-FAC approximation and the effectiveness of our multi-stage influence function. Additionally, case studies on a real-world LLM, dolly-v2-3b, demonstrate its interpretive power, with exemplars illustrating insights provided by multi-stage influence estimates.

1 Introduction

Understanding the relationship between training data and model behavior is essential for building trustworthy machine learning systems. For example, attributing a model's answer in a closed-book question-answering system to a specific Wikipedia article can enhance user trust. Training data attribution (TDA) techniques quantify the contributions of training instances to model behaviors by addressing a counterfactual question: how would the model's behavior change if an example were removed from the training set? Originating from robust statistics [Hampel, 1974] and introduced into deep learning by Koh and Liang [2017], influence function (IF) provides an end-to-end, scalar-valued interpretation of a model's high-level behavior.

While several works on IFs have explored their conceptual framework and applications, they often overlook the scalability of these methods to large-scale neural networks trained on massive datasets. Furthermore, prior analyses predominantly focus on classical architectures, such as feed-forward networks, rather than modern architectures like transformers. Findings of Zhou et al. [2024] indicate that most knowledge in large language models (LLMs) is acquired during pre-training. Consequently, explaining the predictions of fine-tuned models necessitates tracing influence back to the pre-training dataset rather than the fine-tuning dataset. Although Chen et al. [2020] introduced "multi-stage" IFs to address this need, their analysis of NLP models is limited to frozen encoders stacked with linear classifiers, failing to accommodate the full-parameter tuning paradigm prevalent in LLMs.

In this work, we propose an IF-based method to estimate the contribution of pre-training data to the predictions of fine-tuned models, scaling efficiently to LLMs with billions of parameters. Our approach addresses two key challenges.

First, the original IF framework [Koh and Liang, 2017] does not accommodate multi-stage influence when fine-tuning tasks require substantial modifications to the model, such as replacing the unembedding layer. Inspired by Chen et al. [2020], we extend IFs to the multi-stage paradigm ("pre-train then fine-tune"), enabling attribution of downstream predictions of LLMs to pre-training examples.

Second, scaling IFs to LLMs involves overcoming computational bottlenecks related to inverse Hessian-Vector Product (iHVP) and training gradients. For iHVPs, we adopt the Eigenvalue-corrected Kronecker-factored Approximate Curvature (EK-FAC) parameterization [George *et al.*, 2018], as suggested by Grosse et al. [2023]. To address the latter bottleneck, we leverage semantic-similarity-based heuristics to narrow the candidate training samples, avoiding iterations over the entire dataset.

We conduct extensive experiments to evaluate our method. For general influence estimation, we demonstrate the superior scalability of EK-FAC approximations compared to various TDA methods. For our multi-stage IF, we evaluate it on our fact-tracing benchmark and show that it outperforms

*Corresponding author

the single-stage IF [Grosse *et al.*, 2023]. Additionally, we analyze the contributions of MLP and multi-head attention (MHA) parameters to influence estimation, finding that MLP parameters play a proportionally greater role. This insight suggests a practical trade-off, allowing analysis to focus on MLP parameters in large models. Finally, we apply our multi-stage IF on a publicly available instruction-tuned LLM, dolly-v2-3b [Conover *et al.*, 2023], qualitatively explaining its generations based on pre-training data.

In summary, we propose a general framework for efficiently estimating multi-stage influence with the help of EK-FAC parameterization and semantic-similarity-based candidate selection. Our results provide practical insights into resolving influence estimation trade-offs. Furthermore, we demonstrate how TDA approaches can be applied to calibrate the trustworthiness of generative AI systems.

2 Related Work

Training data attribution. Training data attribution (TDA) techniques explain a model's predictions by quantifying the contribution of training data. As noted by Hammoudeh and Lowd [2024], TDA methods can be broadly categorized into retraining-based and gradient-based approaches. *Retraining-based* methods estimate the influence of individual examples by retraining models on random subsets of the training dataset. However, these methods incur high computational costs due to the need for multiple retraining rounds, rendering them impractical for large-scale models and datasets. *Gradient-based* methods, which infer training data influence using gradients, are further divided into dynamic and static approaches. Dynamic estimators, such as TracIn [Pruthi *et al.*, 2020], assess influence by analyzing gradients from intermediate model snapshots captured during training. In contrast, static estimators, including IFs [Koh and Liang, 2017] and representer point selection [Yeh *et al.*, 2018], rely solely on the final model parameters to compute influence.

Influence functions. Despite their utility, IFs exhibit several limitations. First, in terms of *model architecture*, most existing studies focus on traditional architectures such as feed-forward and recurrent networks, with limited exploration of modern architectures like transformers. A recent study by Grosse et al. [2023] extended IF analysis to transformer-based LLMs using EK-FAC approximation. In this work, we also investigate transformer language models.

Second, regarding *scalability*, most research adopts a "matrix-free" approach to avoid the computational costs of explicitly and inverting the Hessian for large models, but with limited success. One prominent strategy involves parametric approximations of the Hessian to enable efficient inverse Hessian computations [Schioppa *et al.*, 2022; Grosse *et al.*, 2023]. Another approach uses iterative stochastic approximation methods, such as LiSSA and Conjugate Gradient, to approximate iHVPs, as introduced by Koh and Liang [2017]. However, our experiments show that these methods fail to yield usable influence estimates within a practical timeframe.

Third, on the *training paradigm*, the original IF proposed by Koh and Liang [2017] is unsuitable for analyzing the influence of pre-training data on a fine-tuned model when it has a different output domain. While Grosse et al. [2023] proposed an efficient method for single-stage training scenarios, it does not address multi-stage paradigms. Chen et al. [2020] introduced multi-stage IF as a generalization of the original IF, but their work is limited to the ELMo [Peters *et al.*, 2018] architecture and fine-tuning scenarios where a classification head is added to a frozen pre-trained encoder. In contrast, we focus on the popular full-parameter fine-tuning paradigm.

3 Background and Notations

3.1 Transformer Architecture

The focus of our work is the transformer language model [Vaswani *et al.*, 2017], which starts with a token embedding, followed by a series of L residual blocks, and ends with a token unembeddings [Elhage *et al.*, 2021] (layer normalization is omitted for brevity). For a sequence t with n tokens, the initial embeddings are $\mathbf{x}_0 = \text{Embed}(t) \in \mathbb{R}^{n \times d}$, hence the start of the residual stream.

At each residual block, the residual stream is first processed by the MHA module: $\tilde{\mathbf{x}}_l = \mathbf{x}_{l-1} + \sum_{h \in H_l} h(\mathbf{x}_{l-1})$, where H_l is the set of attention heads and $\mathbf{x}_{l-1}$ is the input for the l-th ($1 \leq l \leq L$) residual block. The attention pattern for head h is obtained via attention mechanism: $\mathbf{r}^h = \text{Attention}\left(\mathbf{q}^h, \mathbf{k}^h, \mathbf{v}^h\right)$, where $\mathbf{q}^h = \mathbf{x}_{l-1}\left(\mathbf{W}_Q^h\right)^\top$, $\mathbf{k}^h = \mathbf{x}_{l-1}\left(\mathbf{W}_K^h\right)^\top$, $\mathbf{v}^h = \mathbf{x}_{l-1}\left(\mathbf{W}_V^h\right)^\top$. $\mathbf{W}_Q^h, \mathbf{W}_K^h, \mathbf{W}_V^h \in \mathbb{R}^{n_{\text{context}}^H \times d}$ are weights for query, key and value, respectively. The attention pattern is then written into the residual stream: $h(\mathbf{x}_{l-1}) = \mathbf{r}^h \left(\mathbf{W}_O\right)^\top, \mathbf{W}_O \in \mathbb{R}^{d \times n_{\text{context}}^H}$.

The MLP module then processes the output of the MHA and writes back to the residual stream: $\mathbf{x}_l = \tilde{\mathbf{x}}_l + \sigma\left(\tilde{\mathbf{x}}_l \left(\mathbf{W}_I^m\right)^\top\right)\left(\mathbf{W}_O^m\right)^\top$, where $\sigma(\cdot)$ is the element-wise nonlinear activation, $\mathbf{W}_I^m \in \mathbb{R}^{n_{\text{context}}^m \times h}$ and $\mathbf{W}_O^m \in \mathbb{R}^{h \times n_{\text{context}}^m}$ are projection weights. After L residual blocks, the unembeddings produce the final logits: $T(t) = \mathbf{x}_L \mathbf{W}_U^\top$.

3.2 Influence Functions

In this section, we demonstrate the original single-stage IF [Koh and Liang, 2017]. This formulation quantifies the influence of a training example on both model parameters and a measurement of a query sample.

Consider a training dataset $\mathcal{D} = \{z_i\}_{i=1}^N$, where each example z_i represents a token sequence, and the learning task is self-supervised language modeling. The training objective is to minimize the expectation $\mathcal{L}$ of the loss function $\ell(\cdot)$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell(z_i, \boldsymbol{\theta}). \quad (1)$$

The influence of a training sample z on the optimal parameters $\boldsymbol{\theta}^*$ is $\mathcal{I}_{\boldsymbol{\theta}^*}(z_m)$, while the influence on a query z_q is mediated via a differentiable measure, $m(z_q, \boldsymbol{\theta})$, e.g., autoregressive cross-entropy loss of z_q. The influence of z with respect to $\boldsymbol{\theta}^*$ can thus be expressed as:

$$\begin{aligned} \mathcal{I}_m(z, z_q) &= \nabla_{\boldsymbol{\theta}} m(z_q, \boldsymbol{\theta}^*)^\top \mathcal{I}_{\boldsymbol{\theta}^*}(z) \\ &= \nabla_{\boldsymbol{\theta}} m(z_q, \boldsymbol{\theta}^*)^\top \mathbf{H}_{\boldsymbol{\theta}^*}^{-1} \nabla_{\boldsymbol{\theta}} \ell(z, \boldsymbol{\theta}^*), \end{aligned} \quad (2)$$

where $\mathbf{H}_{\boldsymbol{\theta}^*} = \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^*, \mathcal{D})$.

In practice, the Hessian may be singular when the model is not fully converged. To address this, we follow Bae et al. [2022] and replace the Hessian with a damped generalized Gauss-Newton (GGN) matrix [Schraudolph, 2002; Martens, 2020] to ensure positive-definiteness. This modification enables the use of the final model parameters in place of strictly converged parameters $\boldsymbol{\theta}^*$. Additionally, we use the cross-entropy loss to ensure that the loss function is convex with respect to model outputs.

3.3 EK-FAC for Feed-Forward Networks

Naive computation of the GGN ($\mathbf{G}$) or its inverse is computationally prohibitive. To address this, EK-FAC [George *et al.*, 2018] was proposed as an efficient approximation method for computing iHVPs. This approach was subsequently adopted by Grosse et al. [2023] to estimate single-stage influence for LLMs. Structurally, the GGN is approximated as a block-diagonal matrix. Each diagonal block is then approximated using EK-FAC parameterization.

Consider a feed-forward neural network with L layers interleaved with nonlinear activations. Let the parameters be $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_L)$ and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^\top, ..., \boldsymbol{\theta}_L^\top)^\top$, where $\boldsymbol{\theta}_l = \text{vec}(\mathbf{W}_l)$ denotes the vectorized form of $\mathbf{W}_l (l = 1, ..., L)$. The GGN for this network is expressed as:

$$\mathbf{G} = \mathop{\mathbb{E}}_{\substack{(x_n, y_n) \sim \hat{Q}_x \\ y \sim p(y|f_{\boldsymbol{\theta}}(x_n))}} \left[\mathcal{D}\boldsymbol{\theta} \mathcal{D}\boldsymbol{\theta}^\top \right] = \left[\mathbf{G}_{i,j} \right]_{1 \le i,j \le L}, \tag{3}$$

where $\hat{Q}_x$ is the training dataset, $\mathcal{D}\boldsymbol{\theta} = \nabla_{\boldsymbol{\theta}} \log p(y|f_{\boldsymbol{\theta}}(x_n), \boldsymbol{\theta})$, and $\mathbf{G}_{i,j} = \mathbb{E}\left[\mathcal{D}\boldsymbol{\theta}_i \mathcal{D}\boldsymbol{\theta}_j^\top \right]$ is a block of the GGN. Note that the label y is sampled from the model's predictive distribution rather than the training label.

Approximating the GGN as block-diagonal retains only the diagonal blocks: $\mathbf{G} \approx \tilde{\mathbf{G}} = \text{diag}(\mathbf{G}_{1,1}, ..., \mathbf{G}_{L,L})$. Given a vector $\boldsymbol{v}$ with the same dimensions as $\boldsymbol{\theta}$, the damped iHVP can be computed as $\left(\tilde{\mathbf{G}} + \lambda \mathbf{I}\right)^{-1} \mathbf{v} = \text{diag}\left((\mathbf{G}_{1,1} + \lambda \mathbf{I})^{-1} \mathbf{v}_1, ..., (\mathbf{G}_{L,L} + \lambda \mathbf{I})^{-1} \mathbf{v}_L \right)$, where $\mathbf{v}_l$ is the slice of $\mathbf{v}$ corresponding to $\boldsymbol{\theta}_l$.

For each block $\mathbf{G}_{l,l}$, EK-FAC provides a further approximation. Let the inputs and outputs of the l-th layer be $\mathbf{a}_{l-1}$ and $\mathbf{s}_l$, respectively. The gradient $\mathcal{D}\boldsymbol{\theta}_l$ is expressed as $\mathbf{a}_{l-1} \otimes \mathcal{D}\mathbf{s}_l$, where $\otimes$ denotes Kronecker product. $\mathbf{G}_{l,l}$ can thus be approximated using K-FAC as:

$$\begin{aligned} \mathbf{G}_{l,l} &= \mathbb{E}\left[\left(\mathbf{a}_{l-1} \mathbf{a}_{l-1}^\top \right) \otimes \left(\mathcal{D}\mathbf{s}_l \mathcal{D}\mathbf{s}_l^\top \right) \right] \\ &\approx \mathbb{E}\left[\mathbf{a}_{l-1} \mathbf{a}_{l-1}^\top \right] \otimes \mathbb{E}\left[\mathcal{D}\mathbf{s}_l \mathcal{D}\mathbf{s}_l^\top \right] \\ &= \mathbf{A}_{l-1,l-1} \otimes \mathbf{S}_{l,l} = \tilde{\mathbf{G}}_{l,l}. \end{aligned} \tag{4}$$

EK-FAC improves upon K-FAC by incorporating the diagonal variance in the eigenbasis of $\mathbf{A}_{l-1,l-1}$ and $\mathbf{S}_{l,l}$. Denoting these matrices as $\mathbf{A}$ and $\mathbf{S}$ for brevity, their eigendecomposition yields:

$$\begin{aligned} \tilde{\mathbf{G}}_{l,l} &= \mathbf{A} \otimes \mathbf{S} \\ &= (\mathbf{Q_A} \otimes \mathbf{Q_S})(\mathbf{\Lambda_A} \otimes \mathbf{\Lambda_S})(\mathbf{Q_A} \otimes \mathbf{Q_S})^\top, \end{aligned} \tag{5}$$

where $\mathbf{Q_A}$ and $\mathbf{Q_S}$ are eigenvectors, and $\mathbf{\Lambda_A}$ and $\mathbf{\Lambda_S}$ are diagonal matrices of eigenvalues. To account for the diagonal variance, the middle factor is replaced by a new diagonal matrix $\mathbf{\Lambda}$ with its diagonal entries as follows:

$$\begin{aligned} \mathbf{\Lambda}_{ii} &= \mathbb{E}\left[\left(\left(\mathbf{Q_A} \otimes \mathbf{Q_S} \right)^\top \mathcal{D}\boldsymbol{\theta}_l \right)_i^2 \right] \\ &= \mathbb{E}\left[\left(\text{vec}\left(\mathbf{Q_S}^\top \mathcal{D}\mathbf{W}_l \mathbf{Q_A} \right) \right)_i^2 \right]. \end{aligned} \tag{6}$$

Finally, the damped iHVP for the l-th block is computed as:

$$\begin{aligned} (\mathbf{G}_{l,l} + \lambda \mathbf{I})^{-1} \mathbf{v}_l \approx & (\tilde{\mathbf{G}}_{l,l} + \lambda \mathbf{I})^{-1} \mathbf{v}_l \\ \approx & \left(\mathbf{Q_A} \otimes \mathbf{Q_S} \right) \mathbf{\Lambda}_\lambda^{-1} \left(\mathbf{Q_A} \otimes \mathbf{Q_S} \right)^\top \mathbf{v}_l \\ = & \text{vec}\Bigg(\mathbf{Q_S} \Big[\left(\mathbf{Q_S}^\top \bar{\mathbf{V}}_l \mathbf{Q_A} \right) \oslash \\ & \text{unvec}\left(\text{diag}^{-1}\left(\mathbf{\Lambda}_\lambda \right) \right) \Big] \mathbf{Q_A}^\top \Bigg), \end{aligned} \tag{7}$$

where $\mathbf{\Lambda}_\lambda = \mathbf{\Lambda} + \lambda \mathbf{I} (\lambda > 0)$, $\oslash$ denotes element-wise division, $\text{diag}^{-1}(\cdot)$ extracts the diagonal elements of a matrix into a vector, and $\text{unvec}(\cdot)$ converts a vector into a matrix.

4 Method

In this section, we present the design of our multi-stage IF, which attributes the predictions of a fine-tuned LLM to its pre-training data. Additionally, we describe the use of approximation techniques to make the multi-stage IF computationally tractable for LLMs, addressing the trade-off between efficiency and effectiveness.

The objective of the multi-stage IF is to quantify the influence of a pre-training sample z on a test-time query sample x_t. The estimation process consists of two primary steps: candidate selection (Section 4.3) and influence computation (Section 4.2). First, given a query, we filter the pre-training dataset using similarity heuristics for a much smaller subset of training examples as candidates for influence estimation. This step ensures that the selection is focused on training examples that are semantically relevant to the query. In the second step , we compute the influence of the selected candidates on the query, producing a series of influence scores.

4.1 Multi-Stage Influence Function

Before formulating the multi-stage IF, we first review the "pre-train then fine-tune" paradigm. During pre-training, all model parameters are randomly initialized and subsequently updated to fit the distribution of a large-scale corpus.

$$\boldsymbol{\theta}^{\text{pt}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{pt}}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \ell_{\text{pt}}(z_i, \boldsymbol{\theta}), \tag{8}$$

where $\ell_{\text{pt}}(\cdot)$ is the pre-training loss.

During fine-tuning, the model parameters are initialized with $\boldsymbol{\theta}_{\text{pt}}$ and subsequently optimized on a fine-tuning dataset under the cost function $\ell_{\text{ft}}(\cdot)$.

$$\boldsymbol{\theta}^{\text{ft}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{ft}}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{M} \sum_{i=1}^{M} \ell_{\text{ft}}(x_i, \boldsymbol{\theta}). \tag{9}$$

A naive approach to derive multi-stage influence is as follows:

$$\mathcal{I}_m(z, x) = \nabla_{\boldsymbol{\theta}} m(x, \boldsymbol{\theta}^{\text{ft}})^\top \mathbf{H}_{\boldsymbol{\theta}^{\text{ft}}}^{-1} \nabla_{\boldsymbol{\theta}} \ell_{\text{pt}}(z, \boldsymbol{\theta}^{\text{ft}}). \tag{10}$$

This formulation is valid as long as the fine-tuned model shares the same output domain as the pre-trained model. For example, it works for pre-trained language models fine-tuned to follow instructions. However, discrepancies may arise when the final unembedding layer is replaced during fine-tuning. For instance, pre-training typically involves the language modeling task with outputs as logits over a large vocabulary, $f_{\boldsymbol{\theta}^{\text{pt}}}(z) \in \mathbb{R}^{|\mathcal{V}|}$, whereas fine-tuning may adapt the model for binary sequence classification, resulting in outputs $f_{\boldsymbol{\theta}^{\text{ft}}}(x) \in \mathbb{R}^2$. Consequently, the gradient $\nabla_{\boldsymbol{\theta}} \ell_{\text{pt}}(z, \boldsymbol{\theta}^{\text{ft}})$ is formally invalid for the fine-tuned model.

To address this, we propose making the pre-training gradient accessible to the fine-tuned model by establishing a connection between their parameter spaces. Fine-tuning implicitly assumes that the model retains its general capabilities after adaptation. Thus it is reasonable to assume that the fine-tuned parameters do not deviate significantly from pre-trained ones. Under this assumption, fine-tuning can be viewed as minimizing empirical risk on the fine-tuning task in the vicinity of the pre-trained parameters in the parameter space.

Inspired by Chen et al. [2020], we instantiate a quantifiable connection between the pre-trained model and its fine-tuned successor by introducing an additional proximity constraint to the fine-tuning objective in a post-hoc manner. Specifically, we define the proximity as the Euclidean distance between fine-tuned and pre-trained parameters:

$$\boldsymbol{\theta}^{\text{ft}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{ft}}(\boldsymbol{\theta}) + \frac{\alpha}{2} ||\boldsymbol{\theta} - \boldsymbol{\theta}^{\text{pt}}||_2^2, \tag{11}$$

where $\alpha \in \mathbb{R}^+$ is a hyperparameter and $|| \cdot ||_2$ is L2 norm.

Based on this reformulated objective above, the influence of a pre-training example z on a test instance x_t under the measurement $m(\cdot)$ is given by (full proof in Appendix):

$$\begin{aligned} \mathcal{I}_m(z, x_t) = &\nabla_{\boldsymbol{\theta}} m\left(x, \boldsymbol{\theta}^{\text{ft}}\right)^\top \left(\nabla_{\boldsymbol{\theta}}^2 \mathcal{L}_{\text{ft}}\left(\boldsymbol{\theta}^{\text{ft}}\right) + \alpha \mathbf{I}\right)^{-1} \\ &\left(\nabla_{\boldsymbol{\theta}}^2 \mathcal{L}_{\text{pt}}\left(\boldsymbol{\theta}^{\text{pt}}\right)\right)^{-1} \nabla_{\boldsymbol{\theta}} \ell_{\text{pt}}\left(z, \boldsymbol{\theta}^{\text{pt}}\right). \end{aligned} \tag{12}$$

4.2 Practical Implementation for LLMs

A straightforward approach to implementing IFs involves calculating $\mathbf{H}_{\boldsymbol{\theta}}$ and $\mathbf{H}_{\boldsymbol{\theta}}^{-1}$, followed by computing iHVPs via $\mathbf{H}_{\boldsymbol{\theta}}^{-1}\mathbf{v}$. However, for a model with p parameters and N training samples, $\mathcal{O}(Np^2 + p^3)$ operations are required [Koh and Liang, 2017], which is computationally prohibitive when p and N are large. To address this, following Grosse et al. [2023], who approximated single-stage IF with EK-FAC, we adopt a similar approach to approximate the multi-stage IF as described in Section 3.3.

First, we replace Hessians with damped GGNs: $\nabla_{\boldsymbol{\theta}}^2 \mathcal{L}_{\text{ft}}\left(\boldsymbol{\theta}^{\text{ft}}\right) \approx \mathbf{G}_{\text{ft}} + \lambda_{\text{ft}}\mathbf{I}$ and $\nabla_{\boldsymbol{\theta}}^2 \mathcal{L}_{\text{pt}}\left(\boldsymbol{\theta}^{\text{pt}}\right) \approx \mathbf{G}_{\text{pt}} + \lambda_{\text{pt}}\mathbf{I}$. For $\mathbf{G}_{\text{ft}}$, the term α in Equation (12) is absorbed into the damping term λ_{ft}. Consequently, the multi-stage IF (Equation (12)) can be interpreted as the inner product of two preconditioned gradients: $\left(\mathbf{G}_{\text{pt}} + \lambda_{\text{pt}}\mathbf{I}\right)^{-1} \nabla_{\boldsymbol{\theta}} \ell_{\text{pt}}\left(z, \boldsymbol{\theta}^{\text{pt}}\right)$ and $\left(\mathbf{G}_{\text{ft}} + \lambda_{\text{ft}}\mathbf{I}\right)^{-1} \nabla_{\boldsymbol{\theta}} m\left(x_t, \boldsymbol{\theta}^{\text{ft}}\right)$. In practice, we implement multi-stage IF following this interpretation.

Next, we compute EK-FAC factors for $\mathbf{G}_{\text{pt}}$ and $\mathbf{G}_{\text{ft}}$, focusing on the linear components of the model. The EK-FAC factors are precomputed and stored on disk, to be loaded when needed. These include $\mathbf{W}_I^m$ and $\mathbf{W}_O^m$ of MLP modules, $\mathbf{W}_Q^h$, $\mathbf{W}_K^h$ and $\mathbf{W}_V^h$ in each attention head, and $\mathbf{W}_O$ of MHA modules. We exclude unembedding parameters, as the unembedding layer of $\boldsymbol{\theta}^{\text{pt}}$ and $\boldsymbol{\theta}^{\text{ft}}$ may have different output dimensions. We also exclude layer normalization modules, since their parameter count is maginal and they are usually not considered to encode factual knowledge [Grosse *et al.*, 2023].

We focus on autoregressive decoder-only LLMs, the pre-training loss is the cross entropy loss, following Grosse et al. [2023]. For a sequence z with T tokens:

$$\ell_{\text{pt}}(z, \boldsymbol{\theta}^{\text{pt}}) = -\sum_{i=1}^{T} \log p_{\hat{y}|x}(z_i | z_{<i}; \boldsymbol{\theta}^{\text{pt}}), \tag{13}$$

where $p_{\hat{y}|x}$ is the pre-trained model's output distribution.

Computational and Spatial Cost

The one-time cost of preparing EK-FAC factors is considered an overhead amortized across future influence analyses. During influence score computation, for a weight matrix with input dimension d and output dimension p, the cost of computing an iHVP is $\mathcal{O}(d^2p + dp^2)$, followed by an inner product between query iHVP and candidate iHVP at $\mathcal{O}(dp)$. The memory and storage overhead arises from storing eigenvectors $\mathbf{Q}$ and the diagonal entries of $\mathbf{\Lambda}$, resulting in an extra spatial cost of $d^2 + p^2 + dp$.

For example, in the GPT-NeoX [Andonian *et al.*, 2023] architecture, where $d = p$ for query, key, value, and output linear projection weights in MHA modules, the additional spatial cost is 3 times the size of the original weights. For MLP modules, where $d = 4p$ or $d = \frac{1}{4}p$, the additional spatial cost is 5.25 times the size of the original weights.

4.3 Selecting Candidates for Influence Estimation

To identify a few positively influential training examples for a query, a naive approach would compute the influence of every training example on the query. However, this requires gradient computations across the entire dataset, a cost equivalent to one epoch of training. To address this, we narrow down the candidate set using efficient similarity-based heuristics inspired by Grosse et al. [2023].

While Grosse et al. [2023] employed TF-IDF [Ramos, 2003], we adopt an unsupervised K-Nearest Neighbors (KNN) approach based on the embeddings of pre-training documents, similar to the approach of Guo et al. [2021]. The embeddings are generated using Sentence Transformers [Reimers and Gurevych, 2019], and a KNN classifier is constructed over these embeddings. This choice reflects the intuition that training examples with similar semantics to the query are more interpretable and relevant than those based solely on textual overlap [Karpukhin *et al.*, 2020]. Moreover,

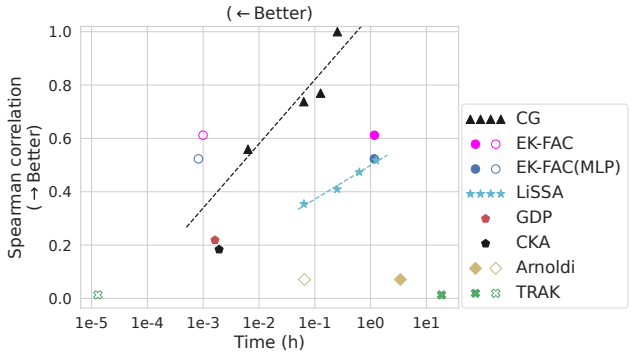


Figure 1: Spearman correlation coefficient of influence scores versus wall-clock time. Hollow markers and solid markers share the same correlation values. Hollow markers only accounts for the time on obtaining pair-wise influence estimates, while solid ones additionally account for the overhead.

Method	Metrics		
	Spearman $\rho \uparrow$	Overhead Time $\downarrow$	Pair-wise Time $\downarrow$
CG	–	0	913.878 s
LiSSA	0.518	0	1.254 h
GDP	0.219	0	5.847 s
CKA	0.184	0	6.444 s
Arnoldi	0.071	3.353 h	235.227 s
TRAK	0.013	18.633 h	**0.047 s**
EK-FAC	**0.612**	1.168 h	3.574 s
EK-FAC(MLP)	<u>0.523</u>	1.141 h	<u>2.645 s</u>

Table 1: Summary of influence estimation quality and runtime. The **best** results are highlighted in bold while <u>second best</u> results are underlined.

this method aligns with the potential application of multi-stage IFs in identifying pre-training documents that serve as grounding knowledge sources.

5 Experiments

This section presents a series of experiments to evaluate our proposed method. First, we assess the scalability of single-stage IFs approximated using EK-FAC compared to various TDA methods in terms of both estimation accuracy and wall-clock runtime on language modeling tasks (Section 5.1). Next, we investigate the validity of the underlying assumption of our multi-stage IF (Section 5.2). Subsequently, we evaluate the effectiveness of the multi-stage IF on a factual knowledge retrieval task (Section 5.3). Finally, we showcase a qualitative case study using the multi-stage IF on an instruction-following LLM (Section 5.4). Our code is public at https://github.com/colored-dye/multi_stage_influence_function.

5.1 Scalability Validation of EK-FAC

This experiment evaluates the effectiveness and efficiency of EK-FAC parameterization in producing influence estimates. For simplicity, we focus exclusively on single-stage language modeling rather than the "pre-train then fine-tune" scenario.

Dataset and Model. We train a custom GPT-NeoX model on The Penn Treebank dataset [Marcus *et al.*, 1993]. The model consists of decoder layers, two attention heads, and a hidden size of 256, comprising approximately 1.18M MLP parameters and 0.39M MHA parameters.

Baselines. We evaluate various TDA methods as baselines. For influence estimation methods relying on iterative iHVP estimation, we include *Conjugate Gradient (CG)* and *LiSSA* following Koh and Liang [2017]. To reduce computational costs, we use *CG* as the ground truth instead of methods like training under the proximal Bregman response objective [Bae *et al.*, 2022] or linear datamodeling scores [Park *et al.*, 2023].

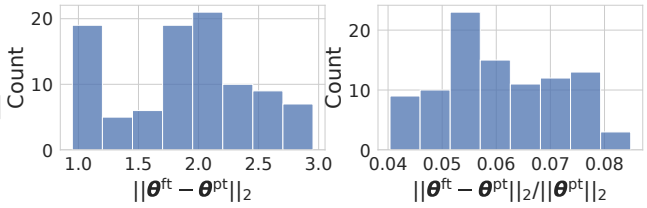
For IF estimation methods based on estimating diagonal entries of the Hessian, we employ IF with *Arnoldi iteration* [Schioppa *et al.*, 2022].

TRAK [Park *et al.*, 2023], a retraining-based baseline, uses a set of models trained on random subsets. We also choose *Gradient dot product (GDP)* [Charpiat *et al.*, 2019; Grosse *et al.*, 2023] and *linear Centered Kernel Alignment (CKA)* [Kornblith *et al.*, 2019] as gradient-similarity-based baselines. *GDP* simply computes the dot product between query gradients and candidate gradients. *CKA* is specifically designed for measuring representation similarity. Here, we use it on query and training gradients.

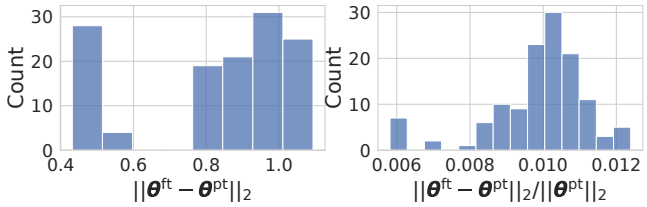
Metrics. We randomly select ten test samples, each paired with 500 candidates sampled from the training split. The primary metric for influence estimation accuracy is the Spearman correlation coefficient (ρ), as ranking quality is our main focus. "Pair-wise runtime" refers to the cost of computing the influence of 500 candidates on a query. "Overhead runtimes" are specific to each baseline: fitting EK-FAC factors for EK-FAC, calculating dominant eigenpairs for *Arnoldi*, or training and gradient featurization for *TRAK*. Spearman correlations and pair-wise runtimes are averaged over ten trials, while overhead runtime is measured once per baseline.

Results. Figure 1 visualizes results, with the best results shown in Table 1. Ideally, a method should achieve high-quality estimates within minimal runtime, corresponding to markers closer to the upper left corner of the figure. Key findings include:

1. EK-FAC achieves the best trade-off between approximation quality and computation cost, with its marker closest to the upper left corner. Furthermore, it resides on a more optimal Pareto frontier than *CG* and *LiSSA*.
2. Ablating influence analysis for MHA parameters does not result in substantial degradation in approximation quality compared to analyzing both MHA and MLP parameters. Despite MHA parameters accounting for 25% of analyzed parameters, they contribute only 14.5% of the total influence, indicating that MLP parameters have a more significant impact.
3. The approximation quality of *CG* and *LiSSA* scales log-linearly with computation time, with *CG* offering a superior Pareto frontier.



(a) BLOOM-560m v.s. BLOOMZ-560m.



(b) Pythia-2.8b v.s. dolly-v2-3b.

Figure 2: Distribution of $||\boldsymbol{\theta}^{\text{ft}} - \boldsymbol{\theta}^{\text{pt}}||_2$ and $||\boldsymbol{\theta}^{\text{ft}} - \boldsymbol{\theta}^{\text{pt}}||_2/||\boldsymbol{\theta}^{\text{pt}}||_2$.

4. Gradient similarity-based methods (*GDP* and *CKA*) are the most efficient baselines but yield low-quality influence estimates.
5. *TRAK*, a representative of retraining-based methods, has the lowest estimation quality and the highest computational cost.
6. *Arnoldi* yields poor estimates at $\sim$120$\times$ the pair-wise compute cost of EK-FAC.

Observation (2) highlights the potential for further approximations in influence estimation. As noted by Grosse et al. [2023], factual associations are primarily localized within MLP modules [Meng *et al.*, 2022], thus MLP modules are also likely to contribute significantly to influence estimation. Our findings support this claim, suggesting that for large-scale models, focusing solely on MLP parameters can significantly reduce computational costs while maintaining useful influence estimates.

5.2 Euclidean Proximity in Practice

As described in Section 4.1, the multi-stage IF relies on the assumption that the fine-tuned parameters are geometrically close to its pre-trained predecessor in the parameter space. To validate this assumption, we conduct an experiment analyzing two pairs of models: BLOOM-560m versus BLOOMZ-560m, and Pythia-2.8b versus dolly-v2-3b. We focus on the linear weights of both the MLP and MHA modules, ignoring bias terms, as the bias parameters only constitute a minor portion of the total parameters.

The results, presented in Figure 2, indicate that the distance between fine-tuned and pre-trained weights ($||\boldsymbol{\theta}^{\text{ft}} - \boldsymbol{\theta}^{\text{pt}}||_2$) accounts for no more than 8% of the L2 norm of the pre-trained weights, $||\boldsymbol{\theta}^{\text{pt}}||_2$. Specifically, for the BLOOM-560m and BLOOMZ-560m pair, $||\boldsymbol{\theta}^{\text{ft}} - \boldsymbol{\theta}^{\text{pt}}||_2^2 = 368.7$, while for the Pythia-2.8b and dolly-v2-3b pair, $||\boldsymbol{\theta}^{\text{ft}} - \boldsymbol{\theta}^{\text{pt}}||_2^2 = 94.5$.

In practice, the damping term of the GGNs is typically very small; in subsequent experiments, we set $\alpha \leq \lambda_{\text{ft}} = 10^{-4}$.

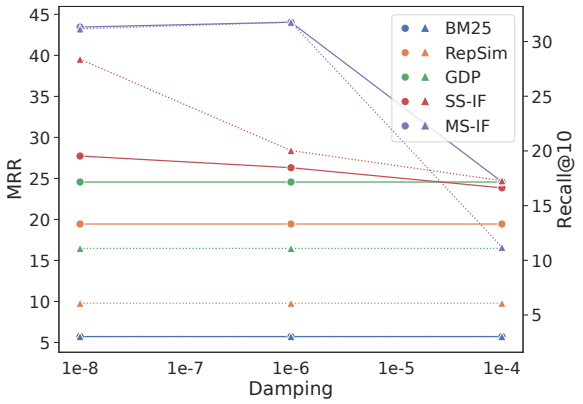


Figure 3: Fact-tracing results (in percentage). Round markers and solid lines denote MRR results, while triangular markers and dotted lines denote Recall@10 results.

For the two model pairs, the additional Euclidean proximity term introduces an extra fine-tuning loss of 0.0184 and 0.0047, respectively. These additional losses are negligible compared to the final training losses of BLOOMZ-560m and dolly-v2-3b, which are 1.6132 and 0.8209, respectively.

5.3 Effectiveness of Multi-Stage Influence Function

In this experiment, we evaluate the effectiveness of our multi-stage IF in identifying pre-training data that significantly influence predictions of a fine-tuned model. To facilitate this evaluation, we establish a benchmark with ground-truth labels for attributing test-set instances to corresponding samples in an attribution set. To enable comparison with the single-stage IF, the chosen task does not require modifying the unembedding layer.

Benchmark Setup. We construct a fact-tracing benchmark based on that of Akyürek et al. [2022], consisting of an attribution set and a test set. Each test set instance corresponds to a fact and is associated with several sentences in the attribution set. The evaluated methods are tasked with correctly retrieving the knowledge sources for each fact.

The attribution set is derived from the T-REx dataset [Elsahar *et al.*, 2018], which aligns knowledge base triples with DBpedia abstracts. Instead of directly using the original dataset, we create a sentence-level subset of T-REx that includes all knowledge base triples represented in the LAMA dataset [Petroni *et al.*, 2019].

The test set is derived from the T-REx split of LAMA. Akyürek et al. [2022] use the test set as cloze-style language modeling samples, as their target models are mT5-based [Xue, 2020] models, which are trained under the masked language modeling "span-corruption" objective. But our target model is an instruction-following autoregressive LLM, so we manually convert the original text completion templates into equivalent question answering formats. For example, a knowledge base triple (X, born_in, Y) originally uses the template "X was born in," which is a text-completion task, while our version reformulates it as "Where was X born?" to

align with a question-answering task, where the model is required to predict the object Y.

Metrics. We evaluate fact retrieval performance using standard information retrieval metrics, including Mean Reciprocal Rank (MRR) and Recall@10. The MRR is defined as $\frac{1}{Q}\sum_{q\in Q}\frac{1}{\text{rank}_q}$, where Q is the test set and rank_q denotes the rank of the first correct knowledge source for query q. Results are averaged over three trials, with each trial sampling 200 test instances.

Baselines. We compare the proposed multi-stage IF (*MS-IF*) against several baseline methods. *BM25* [Robertson *et al.*, 1995], a model-agnostic baseline, retrieves relevant samples based on word-level overlap. Additionally, we include two similarity-based methods: Representation Similarity (*Rep-Sim*) [Caruana *et al.*, 1999], which computes cosine similarity between the hidden states of pre-trained and fine-tuned models, and *GDP*, a simplified multi-stage IF where the Hessian matrices are identity matrices.

The single-stage IF (*SS-IF*) is also evaluated, using only the fine-tuned model. Both *MS-IF* and *SS-IF* adopt the same measurement $m(x, y; \boldsymbol{\theta}) = -\log p_{\boldsymbol{\theta}}(y|x)$, where x and y represent the question and answer tokens, respectively. Both use EK-FAC approximation and analyze both the MLP and MHA modules. Furthermore, as both IFs rely on damping terms, we report their performance under different damping values, ranging from 10^{-8} to 10^{-4}.

Models. We utilize BLOOM-560m [Workshop *et al.*, 2022] and BLOOMZ-560m [Muennighoff *et al.*, 2022], the latter of which has undergone multitask fine-tuning. We choose these models as we are able to verify that their pre-training data encompass most of the knowledge of the fact-tracing benchmark. Specifically, as each knowledge instance is represented as (subject, relation, object) triples, we confirm that 96.81% of the objects are included by the pre-training data of BLOOM(Z)-560m.

Results. As is shown in Figure 3, the multi-stage IF outperforms all baseline methods in terms of both MRR and Recall under smaller damping terms (10^{-6} and 10^{-8}), demonstrating its superior ability to assign higher influence scores to ground-truth knowledge sources. However, the performance gaps between *MS-IF*, *SS-IF*, and *GDP* are relatively small under a large damping term (10^{-4}). This indicates that the choice of the damping term is essential to the effectiveness of IFs, and that both *SS-IF* and *MS-IF* degenerate to *GDP* when the damping terms are large. Moreover, even the best-performing method yields results that are far from perfect retrieval. This discrepancy may stem from approximation errors, the design of our multi-stage IF, or limitations in the language models' suitability for knowledge retrieval tasks.

5.4 Case Study

In addition to the quantitative experiments above, we qualitatively demonstrate the interpretive power of the proposed multi-stage IF through a case study. The analysis is conducted on dolly-v2-3b for the task of factual knowledge attribution. The motivation for this case study stems from possible user concerns regarding whether an LLM-based interactive system's responses are grounded in reliable knowledge sources or are mere hallucinations. To address this, it is reasonable to attribute model-generated outputs to the pre-training data. By inspecting whether the top-influential pre-training texts contain relevant and accurate information, we are able to assess whether the model's responses are properly grounded.

Query	Q: Where did fortune cookies originate? A: The fortune cookie originated in China.
Retrieved Document	Fortune cookies are often served as a dessert in Chinese restaurants in the United States and other Western countries, but are not a tradition in China. [...]As far back as the 19th century, a cookie very similar in appearance to the modern fortune cookie was made in Kyoto, Japan [...]

Table 2: Example of the most influential example from the pre-training dataset for a fact-related query.

For the factual knowledge attribution task, the model is prompted with a question, and a response is generated using greedy decoding. The article with the highest multi-stage influence is identified from the Wikipedia subset of the pre-training corpus using the pipeline described in Section 4. Table 2 presents the results, showing an excerpt of the retrieved article. While the response itself is incorrect, the retrieved article is highly relevant to the queried topic.

6 Conclusions and Limitations

This paper introduces a generalization of the IF to enable the attribution of predictions made by a fine-tuned model to its pre-training data. To enhance the scalability of influence computation, we employ EK-FAC parameterization and a nearest-neighbor-based candidate selection strategy. Experimental results confirm the effectiveness and efficiency of the proposed multi-stage IF, demonstrating its applicability to LLMs with three billion parameters.

While our work enhances the scalability and generality of influence functions, several limitations remain. First, we only analyze backbone MLP and MHA components, excluding contributions from other components such as embeddings, unembeddings, and layer normalization. Extending influence analysis to these components may improve the quality of influence estimates. However, the potential benefit is likely marginal, as they constitute a small proportion of the overall parameters and are not considered to encode knowledge.

Second, our analyses are limited to decoder-only transformer architectures. Extending scalable influence analysis to other architectures, such as encoder-decoder models or diffusion models, could unlock valuable new applications.

Finally, SOURCE is proposed by Bae et al. [2024] as an effective TDA approach, leveraging approximate unrolled differentiation and inherently suited for multi-stage scenario. Their results demonstrate the superior performance of SOURCE compared to influence functions on models like BERT [Devlin, 2018] and GPT-2 [Radford *et al.*, 2019]. Unfortunately, due to the unavailability of its implementation, we were unable to directly compare our multi-stage influence function with SOURCE in this work.

Acknowledgments

This work was partly supported by the National Key Research and Development Program of China under No. 2024YFB3900105, NSFC under No. 62402418, Zhejiang Province's 2025 "Leading Goose + X" Science and Technology Plan under grant No.2025C02034, the Key R&D Program of Ningbo under No. 2024Z115, and the Open Project of Key Laboratory of General Quality Technology and Application of Intelligent Manufacturing Equipment, Ministry of Industry and Information Technology (HK202403532).

References

[Akyürek *et al.*, 2022] Ekin Akyürek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. Towards tracing factual knowledge in language models back to the training data. *arXiv preprint arXiv:2205.11482*, 2022.

[Andonian *et al.*, 2023] Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, Phil Wang, and Samuel Weinbach. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch, 9 2023.

[Bae *et al.*, 2022] 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.

[Bae *et al.*, 2024] Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentation. *arXiv preprint arXiv:2405.12186*, 2024.

[Caruana *et al.*, 1999] Rich Caruana, Hooshang Kangarloo, John David Dionisio, Usha Sinha, and David Johnson. Case-based explanation of non-case-based learning methods. In *Proceedings of the AMIA Symposium*, page 212. American Medical Informatics Association, 1999.

[Charpiat *et al.*, 2019] Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input similarity from the neural network perspective. *Advances in Neural Information Processing Systems*, 32, 2019.

[Chen *et al.*, 2020] Hongge Chen, Si Si, Yang Li, Ciprian Chelba, Sanjiv Kumar, Duane Boning, and Cho-Jui Hsieh. Multi-stage influence function. *Advances in Neural Information Processing Systems*, 33:12732–12742, 2020.

[Conover *et al.*, 2023] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[Devlin, 2018] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Elhage *et al.*, 2021] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[Elsahar *et al.*, 2018] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[George *et al.*, 2018] 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.

[Grosse *et al.*, 2023] 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, 2023.

[Guo *et al.*, 2021] Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, 2021.

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

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

[Karpukhin *et al.*, 2020] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

[Koh and Liang, 2017] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[Kornblith *et al.*, 2019] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International*

conference on machine learning, pages 3519–3529. PMLR, 2019.

[Marcus *et al.*, 1993] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

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

[Meng *et al.*, 2022] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

[Muennighoff *et al.*, 2022] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*, 2022.

[Park *et al.*, 2023] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning (ICML)*, 2023.

[Peters *et al.*, 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Petroni *et al.*, 2019] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[Pruthi *et al.*, 2020] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

[Radford *et al.*, 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[Ramos, 2003] Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[Reimers and Gurevych, 2019] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[Robertson *et al.*, 1995] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[Schioppa *et al.*, 2022] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186, 2022.

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

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Workshop *et al.*, 2022] BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[Xue, 2020] L Xue. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[Yeh *et al.*, 2018] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[Zhou *et al.*, 2024] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.