Accountability Attribution: Tracing Model Behavior to Training Processes

Anonymous Authors[1]

Abstract

As foundation models (FMs) advance toward artificial general intelligence with increasingly complex training pipelines—involving pretraining, fine-tuning rounds, and subsequent adaptation or alignment—ensuring their reliable and responsible development becomes critical. A key challenge in this context is accountability: when a deployed model exhibits concerning or beneficial behaviors, which training stage is responsible, and to what extent? We pose the problem of **accountability attribution**, which aims to trace model behavior back to specific stages of the training process, enabling transparency and auditability essential for responsible AI development. To address this, we propose a general framework that answers counterfactual questions about stage effects: *how would the model's behavior have changed if the updates from a training stage had not been executed?*. Within this framework, we introduce estimators based on first-order approximations that efficiently quantify the stage effects without retraining. Our estimators account for both the training data and key aspects of optimization dynamics, including learning rate schedules, momentum, and weight decay. Through experiments on high-stakes domains like medicine and finance, we demonstrate that our approach identifies training stages accountable for specific behaviors, offering a practical tool for ensuring reliable and responsible foundation model development.

1. Introduction

Modern foundation models (FMs) are developed through multiple training stages, including pretraining, domain-specific fine-tuning, and subsequent downstream adaptation or alignment, each encompassing numerous parameter update steps shaped by distinct data and optimization dynamics (Lopez-Paz & Ranzato, 2017; Kornblith et al., 2019; Raghu et al., 2019; He et al., 2022; Chen et al., 2020; Radford et al., 2019; Ouyang et al., 2022; Hu et al., 2022). While this modular structure has become central to achieving state-of-the-art performance and emergent capabilities, it complicates a critical question of accountability: when a foundation model exhibits harmful, beneficial, or surprising behavior, which stage of the training process bears responsibility? This question, lying at the intersection of explainability, causality, and learning dynamics, remains largely underexplored in current practice. As foundation models are increasingly deployed in high-stakes settings like medicine and finance, answering this question becomes essential for model debugging, auditing, and enforcing accountability to properly credit or blame the appropriate training stages - a key requirement for reliable and responsible AI development.

We formulate the **accountability attribution** problem to address this challenge: tracing model behavior to stages of the training process that shaped it. This problem relates to, yet remains distinct from, three research directions. First, causal responsibility analysis (Chockler & Halpern, 2004; Halpern & Pearl, 2005; Triantafyllou et al., 2021) provides formal definitions of blame and responsibility through structural causal models but has primarily focused on discrete decision-making settings at small scales, such as two people throwing rocks at a bottle (Halpern & Pearl, 2005), making it challenging to apply to high-dimensional, sequential processes like training foundation models. Second, research on learning dynamics (Ren et al., 2022; Ren & Sutherland, 2025; Park et al., 2024) investigates how model parameters evolve during training and their consequent impact on test performance, revealing phenomena such as phase transitions or representation formation (Park et al., 2024). However, this research typically aims at descriptive understanding rather than attributing credit or blame to specific training stages. Third, data attribution methods (Koh & Liang, 2017; Ghorbani & Zou, 2019; Ilyas et al., 2022; Pruthi et al., 2020; Bae et al., 2024; Wang et al., 2025) trace model behavior to individual data points. While these methods can assign data-level accountability, they primarily study the "average model" expected to be trained from a given dataset, often overlooking the actual training process (Koh & Liang, 2017).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on ICML 2025 Workshop on Reliable and Responsible Foundation Models. Do not distribute.

Their assumptions and simplifications (e.g., convexity, convergence, permutation invariance) limit their applicability to training stage-specific attribution. Although recent work has extended to consider training processes (Bae et al., 2024; Wang et al., 2025), these approaches remain fundamentally data-centric and assume basic SGD optimizers, failing to account for the impact of complex optimization dynamics in practice.

To address these gaps, we propose a general framework for accountability attribution that explicitly analyzes the training process as a sequence of interventions. Our framework builds on the potential outcomes formalism (Rubin, 1974; 2005), enabling counterfactual queries about the effect of training stages: *how would the model's behavior have changed if the updates from a specific training stage had not been executed?* The framework focuses on estimating the causal effects of training stages, which are defined as sets of model update steps determined by both training data and optimization dynamics, including influences from learning rate schedules, momentum, and weight decay. This approach provides model-specific attribution results by considering the complete training process.

We instantiate this framework using first-order approximations that estimate the effect of training stages. Our estimators are both efficient and flexible: they avoid retraining, scale to deep foundation models, and yield reusable "stage embeddings" that capture the essential influence patterns of each training stage. These stage embeddings only need to be computed once during training and can be applied to analyze accountability for model behavior on any test input or performance function. We refer to the estimated performance effect as the *Accountability Attribution Score (AA-Score)* of the training stage.

Through experiments on vision and language foundation models, we show that our method reliably identifies training stages that are responsible for critical model behaviors—including the introduction of spurious correlations, the learning of domain generalization, or the degradation from noisy labels. These results position accountability attribution as a practical and principled tool for ensuring the reliable and responsible development of foundation models through enhanced transparency and auditability. Our contributions are summarized as follows:

- We pose and formulate the **accountability attribution problem** as tracing model behavior to stages of the training process, enabling responsible development of foundation models.
- We propose a **general framework** for accountability attribution based on the potential outcomes formalism, enabling counterfactual queries about the effect of training stages.
- We derive efficient **estimators** within this framework that quantify stage effects while accounting for **optimization dynamics** including learning rate schedules, momentum, and weight decay.
- We demonstrate the framework's **practical utility** across diverse foundation model settings, showing that it uncovers influential stages responsible for beneficial and harmful model behaviors.

2. Related work

Responsibility and causal analysis The assessment of responsibility is a fundamental challenge in practice that often requires careful consideration of causality (Chockler & Halpern, 2004). Structural causal models serve as powerful tools for formalizing this concept (Halpern & Pearl, 2005; Pearl, 2009), enabling precise definitions of blame and responsibility through counterfactual dependence (Halpern & Kleiman-Weiner, 2018). In the context of AI, these frameworks have been extended to analyze multi-agent settings (Triantafyllou et al., 2021) and human-AI collaboration (Qi et al., 2024). While these formalisms provide valuable perspectives on responsibility attribution, they typically focus on relatively simple problems in small settings with enumerable outcomes, e.g., two people throwing rocks at a bottle. Applying them directly to the high-dimensional, sequential process of training deep AI models presents significant challenges. For such complex processes of AI model training, a notable related work by (Lesci et al., 2024) employs a potential outcome causal framework to study memorization. While our work shares their goal of using causal reasoning and the potential outcome framework, we focus specifically on attributing model behavior to training stages and determining their accountability.

Learning dynamics describes how AI models usually learn new knowledge by updating their parameters via gradient-based optimization. It links changes in the model's parameters or predictions over time, to the gradients generated by learning specific examples (Ren et al., 2022). Through analyzing the learning dynamics, interesting phenomena during training has been explained, such as the "zig-zag" learning path (Ren et al., 2022), the "squeezing effect" of LLM finetuning (Ren & Sutherland, 2025), and the formation of compositional concept spaces (Park et al., 2024). Our work complements these studies by providing a method to quantify the contribution of training stages to the final outcome, potentially helping to explain the mechanisms behind observed dynamic phenomena. Our method is also more quantitative and can be efficiently applied to different test data or performance metrics through the use of stage embeddings.

Data attribution aims to trace model behavior back to the training data instances. Classical approaches like influence functions (Cook & Weisberg, 1980; Koh & Liang, 2017), Data Shapley (Ghorbani & Zou, 2019), and retraining methods like Datamodels (Ilyas et al., 2022) analyze accountability from the data perspective. These methods study an "average model" expected to be trained from a given dataset and thus are limited to analyze single model instances produced by a specific training process. Moreover, they often rely on assumptions such as convergence, convexity, or permutation invariance of traning data that limit their applicability to non-convergent models, multi-stage training processes, and permutation of training data. There is a line of data attribution research that specifically examines the training process, including methods like TracIn (Pruthi et al., 2020), approximate unrolled differentiation (Bae et al., 2024), and Data Value Embedding (DVEmb) (Wang et al., 2025). These process-based approaches better capture temporal dependencies by tracing influence along the optimization trajectory. The most closely related work to ours is DVEmb, which traces training example influence along the optimization trajectory using first-order approximations for leave-one-out (LOO) counterfactuals. Our work differs by analyzing the specific counterfactual of training stages instead of only data points, and considering a more complete, practical optimization process incorporating learning rate schedules, momentum, and weight decay. Ours provides a distinct perspective that incorporates the training data and also the optimizer state for each update step.

3. Preliminaries: optimization dynamics and causal analysis

3.1. Optimization dynamics

Let $p(\boldsymbol{x};\boldsymbol{\theta})$ be a model on instances $\boldsymbol{x} \in \mathbb{R}^d$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^p$. Training starts from an initial state $\boldsymbol{\xi}_0 = (\boldsymbol{\theta}_0, \boldsymbol{v}_0)$, where $\boldsymbol{v}$ is the velocity for momentum-based optimizers, typically the zero vector when initialized. Training proceeds for K steps using a dataset $\mathcal{D}$, typically partitioned into ordered batches $\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_{K-1}$. At each step k (from 0 to $K-1$), the parameters and velocity are updated based on a batch $\mathcal{B}_k$, a training loss function $\mathcal{L}$, a learning rate η_k, a momentum factor μ, and a weight decay factor λ. This sequence of updates defines the observed training state trajectory $\boldsymbol{\xi}_k = (\boldsymbol{\theta}_k, \boldsymbol{v}_k)$ for $k = 0, \ldots, K$ including the parameters $\boldsymbol{\theta}$ and velocity $\boldsymbol{v}$. The specific update rules considered in this paper is SGD with momentum and weight decay (Sutskever et al., 2013), with the implementation closely following modern deep learning frameworks like PyTorch (Paszke et al., 2017):

$$G_k = \sum_{\boldsymbol{x} \in \mathcal{B}_k} \nabla \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{x}) \tag{1}$$

$$G_k{}^{wd} = G_k + \lambda \boldsymbol{\theta}_k \tag{2}$$

$$\boldsymbol{v}_{k+1} = \mu \boldsymbol{v}_k + G_k{}^{wd} \tag{3}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{v}_{k+1} \tag{4}$$

3.2. Causal analysis framework: potential outcomes

To formally analyze accountability, we utilize the **potential outcomes** framework (Rubin, 1974; 2005), which provides a rigorous foundation for describing the causal effect of an intervention (**treatment**) on a target quantity (**outcome**).

Let $T \in \{0, 1\}$ denote a binary treatment assignment variable, representing the intervention to be studied, e.g., whether a training stage has happened during model training ($T = 1$) or not ($T = 0$). The outcome variable Y represents our quantity of interest affected by the treatment, such as the model's final performance. To properly define the causal effect of T on Y, we must consider two scenarios: the outcome when $T = 0$ and when $T = 1$. For a deployed model, only one of these scenarios will be observed, and the other will be counterfactual. The potential outcomes framework provides the formal notation to represent both scenarios.

Definition 3.1. *The **potential outcome** $Y(T)$ represents the value that the outcome variable Y would attain if the treatment assignment were set to T.*

Definition 3.2. *The **causal effect** τ of the treatment, also known as the individual treatment effect (ITE), is defined as the difference between the potential outcomes under treatment and control: $\tau = Y(1) - Y(0)$.*

A fundamental challenge in causal inference is that we can only observe one outcome Y—normally the one corresponding to the treatment actually received (Holland, 1986). The **consistency property** (Cole & Frangakis, 2009) establishes the relationship between this observed outcome Y and the potential outcome under the received treatment $Y(T)$, i.e., $Y(1) = Y$. To estimate τ, we must develop methods to estimate the unobserved counterfactual outcome $Y(0)$.

4. A framework for accountability attribution

4.1. Problem formalization: causal effect of a training stage

We define the accountability attribution problem as the causal effect of a training stage on the final model performance. Building upon the general causal framework in §3.2, we present our framework for accountability attribution by specifying the treatment and outcomes for the problem. We define the treatment as whether the training

stage has happened or not. Formally, for a model training process that evolves from $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_K$, let $S = \{t_1, \ldots, t_s\}$ be the time indices of a training stage involving training steps $t_i \in \{0, \ldots, K-1\}$ for all $i \in \{1, \ldots, s\}$. The treatment $T_S \in \{0, 1\}$ indicates whether the model updates at steps in S are been executed ($T_S = 1$) or all steps in S are skipped ($T_S = 0$). At each time step k, we define the potential outcome of the model state under the treatment as $\boldsymbol{\xi}_k(T_S) = (\boldsymbol{\theta}_k(T_S), \boldsymbol{v}_k(T_S))$.

- The observed (treated) trajectory corresponds to $T_S = 1$: $\boldsymbol{\xi}_k(1_S) = \boldsymbol{\xi}_k$ by the consistency property.
- The counterfactual (controlled) trajectory, where the stage S is skipped, is denoted $\boldsymbol{\xi}_k(0_S)$. This trajectory evolves by executing the standard update for $k \notin S$ and skipping the update for $k \in S$. That is, if $k \in S$, then $\boldsymbol{\xi}_{k+1}(0_S) = \boldsymbol{\xi}_k(0_S)$.

We use a performance function $\gamma(\boldsymbol{x}, \boldsymbol{\theta})$ to quantify the model's performance on an instance $\boldsymbol{x}$ at a given state $\boldsymbol{\theta}$, for example, the log-likelihood $\log p(\boldsymbol{x}; \boldsymbol{\theta})$. For each time k, we define the outcome variable under treatment T_S as $Y_k(T_S) = \gamma(\boldsymbol{x}, \boldsymbol{\theta}_k(T_S))$.

Finally, the accountability attributed to the training stage S is then the causal effect of the treatment on the performance function γ at the final time step K:

$$\tau_{K,S} = Y_K(1_S) - Y_K(0_S) = \gamma(\boldsymbol{x}, \boldsymbol{\theta}_K(1_S)) - \gamma(\boldsymbol{x}, \boldsymbol{\theta}_K(0_S)). \tag{5}$$

To solve the accountability attribution problem, any estimator for the causal effect $\tau_{K,S}$ can be plugged in to our framework. In the following sections, we present our estimator using interpolation and a first-order Taylor expansion, which results in the AA-Score of a training stage.

4.2. Estimating effects of training stages

To build towards the estimation of the effect of a training stage, i.e., the AA-Score, we first consider the special case of the treatment only including a single step t, i.e., $S = \{t\}$. We write T_S as T for simplicity to refer to the treatment of step t. We introduce an interpolation parameter $\epsilon \in [0, 1]$ that defines a continuous path between the observed state $\boldsymbol{\xi}_k(1)$ at $\epsilon = 1$ and the counterfactual state $\boldsymbol{\xi}_k(0)$ at $\epsilon = 0$. Let $\boldsymbol{\xi}_k(\epsilon)$ denote this interpolated state.

For step k up to t, $\boldsymbol{\xi}_k(\epsilon) = \boldsymbol{\xi}_k$. At step t, the state $\boldsymbol{\xi}_t$ is the same for both paths. The difference due to executing step t is $\boldsymbol{\xi}_{t+1} - \boldsymbol{\xi}_t$, for both $\boldsymbol{\theta}$ and $\boldsymbol{v}$. We define the interpolated state after step t as:

$$\boldsymbol{\theta}_{t+1}(\epsilon) = \boldsymbol{\theta}_t + \epsilon(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \epsilon\eta_t \boldsymbol{v}_{t+1} \tag{6}$$

$$\boldsymbol{v}_{t+1}(\epsilon) = \boldsymbol{v}_t + \epsilon(\boldsymbol{v}_{t+1} - \boldsymbol{v}_t) \tag{7}$$

For step k after $t+1$, $\boldsymbol{\xi}_k(\epsilon)$ evolves from $\boldsymbol{\xi}_{t+1}(\epsilon)$ using standard optimization dynamics. This construction ensures $\boldsymbol{\xi}_k(\epsilon = 0) = \boldsymbol{\xi}_k(T = 0)$ and $\boldsymbol{\xi}_k(\epsilon = 1) = \boldsymbol{\xi}_k(T = 1)$.

Our main result is an estimator for the causal effect $\tau_{K,t}$ by first-order Taylor expansion at the observed path. To derive that estimator, we first show an intermediate result of estimating the causal effect on the state $\boldsymbol{\xi}$, where we start from the difference between the observed and counterfactual states at step $t+1$ and propogate the difference step by step to the final step K.

Estimator 4.1 (Single step effect on the training state)**.** *Let $\tau^{\boldsymbol{\xi}}_{K,t} = \boldsymbol{\xi}_K(1) - \boldsymbol{\xi}_K(0)$ be the causal effect on the final state by step t, $t \in \{0, \ldots, K-1\}$. Define the initial statedifference at step $t+1$ as:*

$$\boldsymbol{w}_{t+1,t} = \begin{pmatrix} \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \\ \boldsymbol{v}_{t+1} - \boldsymbol{v}_t \end{pmatrix} = \begin{pmatrix} -\eta_t \boldsymbol{v}_{t+1} \\ \boldsymbol{v}_{t+1} - \boldsymbol{v}_t \end{pmatrix} \tag{8}$$

For all steps $k \in \{t+1, \ldots, K-1\}$, define the one-step propagator matrix as:

$$\mathbf{M}_k = \begin{pmatrix} \mathbf{I} - \eta_k(H_k + \lambda \mathbf{I}) & -\eta_k \mu \mathbf{I} \\ H_k + \lambda \mathbf{I} & \mu \mathbf{I} \end{pmatrix} \tag{9}$$

where $H_k = \sum_{\boldsymbol{x} \in \mathcal{B}_k} \nabla^2 \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{x})$ is the Hessian of the training loss $\mathcal{L}$ evaluated at the observed $\boldsymbol{\theta}_k$. The matrix $\mathbf{M}_k$ connects the difference in state at step k and step $k+1$ as $\boldsymbol{w}_{k+1,t} = \mathbf{M}_k \boldsymbol{w}_{k,t}$. Define the overall propagator matrix $\mathbf{P}^{((t+1)\to K)}$ as:

$$\mathbf{P}^{((t+1)\to K)} = \begin{cases} \prod_{k=K-1}^{t+1} \mathbf{M}_k & \text{if } 0 \le t < K-1 \\ \mathbf{I} & \text{if } t = K-1 \end{cases}$$

Then, a first-order estimator of $\tau^{\boldsymbol{\xi}}_{K,t}$ is:

$$\hat{\tau}^{\boldsymbol{\xi}}_{K,t} = \boldsymbol{w}_{K,t} = \mathbf{P}^{((t+1)\to K)} \boldsymbol{w}_{t+1,t} \tag{10}$$

Now by plugging in the performance function γ into the estimator of effect on the training state, we can derive an estimator for the causal effect on the performance.

Estimator 4.2 (Single step effect on the model performance)**.** *Let $\tau_{K,t} = Y_K(1) - Y_K(0)$ be the causal effect of step t on a performance function $\gamma(\boldsymbol{x}, \boldsymbol{\theta}_K) = Y_K$. Let $\mathbf{P}^{((t+1)\to K)} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{pmatrix}$ be the overall propagator matrix defined in Estimator 4.1. Let $E_t = \mathbf{P}_{11}(-\eta_t \boldsymbol{v}_{t+1}) + \mathbf{P}_{12}(\boldsymbol{v}_{t+1} - \boldsymbol{v}_t)$ be the difference in parameters (first block of the state difference vector $\boldsymbol{w}_{K,t}$). A first-order estimator of $\tau_{K,t}$ is:*

$$\hat{\tau}_{K,t} = \nabla_{\boldsymbol{\theta}} \gamma(\boldsymbol{x}, \boldsymbol{\theta}_K)^\top E_t \tag{11}$$

where the gradient $\nabla_{\boldsymbol{\theta}} \gamma$ is evaluated at the observed final parameters $\boldsymbol{\theta}_K$.

The detailed derivation of Estimators 4.1 and 4.2 is deferred to App. A.1. Importantly, we see that eq. (11) is the product of two parts. The second part E_t depends on training dynamics and the treatment step t but is independent of the test instance $\boldsymbol{x}$ or the performance function γ. This allows E_t to be computed during training and reused as an "embedding" to efficiently estimate the performance effect on any new data instance by doing a dot product. The computational considerations and efficient implementation strategies for computing E_t are discussed in §4.3.

Once we establish the estimator of the causal effect of a single step, we can extend it to estimate the effect of an entire training stage. We show in the following that the first-order estimator of the total effect of a training stage is the sum of the effects calculated individually for each step in the stage.

Estimator 4.3 (Effect of a training stage)**.** *Let $S = \{t_1, \ldots, t_s\}$ be a training stage with steps $t_i \in \{0, \ldots, K-1\}$ for all $i \in \{1, \ldots, s\}$. Let $\tau^{\boldsymbol{\xi}}_{K,S}$ and $\tau_{K,S}$ denote the causal effects of stage S on the training state and performance, respectively. Let $\hat{\tau}^{\boldsymbol{\xi}}_{K,t_i}$ and $\hat{\tau}_{K,t_i}$ be the effect estimators of step t_i as defined in Estimators 4.1 and 4.2. A first-order estimator of $\tau^{\boldsymbol{\xi}}_{K,S}$ and $\tau_{K,S}$ are given by the sum of the effect estimators for each step in S:*

$$\hat{\tau}^{\boldsymbol{\xi}}_{K,S} = \sum_{t_i \in S} \hat{\tau}^{\boldsymbol{\xi}}_{K,t_i}, \quad \hat{\tau}_{K,S} = \sum_{t_i \in S} \hat{\tau}_{K,t_i} \tag{12}$$

The proof, based on the linearity of the first-order approximation derived from a multi-parameter Taylor expansion, is provided in App. A.2. The estimated performance effect $\hat{\tau}_{K,S}$ in eq. (12) will be the AA-Score of the training stage S.

4.3. Computational considerations

To estimate the effect of a training stage, we need to compute E_t for each step in the stage as outlined in Estimators 4.1 and 4.2. Althought we only need to do this computation once along the training process, the direct computation of E_t presents significant computational challenges. These primarily arise from the manipulation of the propagator matrices $\mathbf{M}_k$ and $\mathbf{P}^{((t+1)\to K)}$, which have size $2p \times 2p$, and the Hessian computation which is $O(p^2)$, where p is the dimension of the parameter vector $\boldsymbol{\theta}$.

Complexity of full propagation The propagation matrix $\mathbf{M}_k$ is of size $2p \times 2p$. Computing the overall propagator $\mathbf{P}^{((t+1)\to K)}$ involves approximately $K - t$ matrix-matrix multiplications. If each $\mathbf{M}_k$ is explicitly formed, each such multiplication costs $O((2p)^3) = O(p^3)$. Thus, forming $\mathbf{P}^{((t+1)\to K)}$ for a single step t can be $O((K-t)p^3)$. An iterative algorithm can be used to compute all E_t by updating a backward product, e.g., first computes $\mathbf{P}^{((K-1)\to K)}$ and then uses it to compute E_{K-1}. Then, update $\mathbf{P}^{((K-1)\to K)}$ to $\mathbf{P}^{((K-2)\to K)}$ by multiplying it with $\mathbf{M}_{K-2}$ and uses it to compute E_{K-2}, etc. The per-step cost in the backward pass involves a matrix-matrix product, leading to an overall complexity that can be roughly $O(Kp^3)$ or $O(K|\mathcal{B}|p^2)$ if Hessian-vector products are used efficiently within the matrix multiplication. The storage for $\mathbf{P}^{((t+1)\to K)}$ itself is $O(p^2)$.

Hessian approximation The computation of $\mathbf{M}_k$ requires the Hessian of the training loss $H_k = \sum_{\boldsymbol{x} \in \mathcal{B}_k} \nabla^2 \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{x})$. Forming this $p \times p$ matrix is typically intractable. In practice, we approximate H_k using the Generalized Gauss-Newton (GGN) matrix: $H_k \approx \sum_{\boldsymbol{x} \in \mathcal{B}_k} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{x})^\top$, which is common in the literature (Martens, 2020). The advantage of the GGN (and other outer-product approximations) is that its product with a vector $\boldsymbol{z}$ (i.e., $H_k \boldsymbol{z}$) can be computed efficiently without explicitly forming H_k: $\left(\sum \boldsymbol{g}\boldsymbol{g}^\top\right)\boldsymbol{z} = \sum \boldsymbol{g}(\boldsymbol{g}^\top \boldsymbol{z})$. This reduces the cost of applying H_k from $O(p^2)$ to $O(|\mathcal{B}|p)$. This efficiency is crucial when computing the action of $\mathbf{M}_k$ on a vector.

Layer-wise computation (approximation) A common heuristic to reduce dimensionality is to restrict the computation of effect to the parameters of each layer l (with dimension p_l) separately and then aggregate the effects. This effectively assuming the independence between effects of different layers, and it is common in the literature with influence analysis of large models (Grosse et al., 2023). This will reduce to the computation of per-layer effects embeddings E^l_t, and the overall complexity will also become $O(K|\mathcal{B}|\sum_l p_l^2)$. A more aggressive approximation is to only consider the effect of a subset of layers or even a single layer, e.g., the last layer for prediction. This will reduce the complexity to $O(K|\mathcal{B}|p_l^2)$. Our empirical observations (and those in related literature (Koh & Liang, 2017; Barshan et al., 2020)) suggest that often only computations focused on the last layer yield reasonably stable or interpretable results when such drastic approximations are made.

5. Experiments

5.1. Datasets and experiment settings

We consider four datasets: MNIST (lec), CELEBA (Liu et al., 2015), for image classification, and CIVILCOMMENTS (Borkan et al., 2019) for text toxicity classification. We use the Wilds benchmark (Koh et al., 2021) for the CELEBA and CIVILCOMMENTS datasets. For each dataset, we employ model architectures appropriate to the task. We start with simple Multi-Layer Perceptrons (MLPs) on MNIST to facilitate detailed analysis and direct comparison with retraining. For CELEBA, we employ standard ResNets (He et al., 2016) to assess our method on more complex image recognition tasks. For the CIVIL-

COMMENTS dataset, we fine-tune a pre-trained Transformer model, specifically a GPT-2 (Radford et al., 2019) from the Huggingface library (Wolf et al., 2019), to evaluate the influence of training steps in the context of fine-tuning language models. In our experiments, we use the log-likelihood as the performance function γ and estimate the performance effect $\hat{\tau}_{t,K}$ as the AA-Score of the stage. Therefore, a **positive** $\hat{\tau}_{t,K}$ indicates that the training stage contributes to a **higher log-likelihood**, i.e., the stage is **beneficial** to the model's performance. We train these models and implement our estimators on a server with 64 cores and one NVIDIA A100 GPU with 40G memory. Details of datasets, model architectures, and hyperparameters are deferred to App. B.1.

5.2. Accountability attribution on MNIST

We start with accountability attribution for MLPs trained on MNIST. Since the setting is simple, we can perform model retraining without certain training stages to get the counterfactual effect of that stage as the gold-standard. We then use our estimators to estimate that stage's effect and compare the results. We consider several semi-synthetic settings to demonstrate the utility of our method. Major results are shown in Fig. 1, with experiment details and additional results in App. C.

Capture influence of optimization parameters We start by showing our AA-Score considers the effect of training stages based on the optimization parameters that affect the training process, including the learning rate (lr), momentum, and weight decay (wd). We consider cases where we vary these parameters to separate the training into two stages with different optimization parameters, and analyze the stage effects and observe their influence. When we vary each parameter, we keep the other parameters the same across the two stages. We show the results in Fig. 1 (a-d). In (a), we show the baseline case of one stage, all three parameters stay the same, with lr=0.01, momentum=0.9, and wd=1e-5. These are the common settings for training MLPs on MNIST and the parameters for the first stage for all other settings. In (b), we set the lr to 0.001 in stage 2. In (c), we set the momentum to 0.1 in stage 2. In (d), we set the weight decay to 0.1 in stage 2. We see that as the lr decreases, the AA-Score decreases as well. This is as expected because the second stage with smaller lr have less impact on the model's parameters. We also see that as the momentum decreases, the AA-Score shows similar behavior to the lr. We also see that as the weight decay increases, steps scores in the second stage become closer to zero, for both positive and negative scores. This is because the meaningful learning signals come from the data are less significant with larger weight decay. These results work as sanity checks for our method, as they show that our method can capture the effect of optimization parameters on the model's performance quantitatively.

Detect an influential stage Next, we consider the case of detecting an influential training stage, for simplicity, we consider a stage with one update step and apply our Estimator 4.2 to estimate the effect of the stage. Specifically, we exclude all instances of a specific digit (e.g., digit '4') from the training set. Then, during a single training step, we insert a data point of the digit '4' (an influential stage with one update step). We estimate the effect of all training steps, and show that the inserted step will have a high score on the model's performance on the digit '4' (tested on the same image and similar '4's), demonstrating AA-Score can identify stages processing influential updates to the model. We show the results in Fig. 1 (e).

Capture a negative stage caused by mislabeled data We then consider the case of capturing a stage that have negative effect on the model's performance, e.g., due to mislabeled data. Specifically, we modify labels of a small percentage of data points in the training set. We then estimate the effect of all training steps, and show that the stage with steps processing mislabeled data will have negative scores regarding the model's final performance on a test set, demonstrating AA-Score can capture negative effects of training stages. We show the results in Fig. 1 (f).

Multi-stage training with distributional shifts Multi-stage training with distributional shifts can occur naturally in scenarios like continual learning, domain adaptation, and model fine-tuning. Understanding how each stage with a different data distribution affects the final model is crucial for diagnosing issues like catastrophic forgetting or identifying when spurious correlations are learned. To mimic multi-stage training and distributional shifts, we will train on MNIST in three stages. Stage 1: standard MNIST. Stage 2: MNIST images rotated by a degree (e.g., 45 degrees). Stage 3: MNIST images rotated by another degree (e.g., 90 degrees). We will then evaluate the effect of training steps from each stage on the model's performance on test sets corresponding to each of these three distributions (original, 45-degree rotated, 90-degree rotated). This will help understand how and when the model adapts to or forgets information from different training phases. We observe that the effect of each stage is the highest on the test set with the same distribution as the stage, as expected. We show the results in Fig. 1 (g-i).

For the cases of insertion, mislabeled data, and distributional shifts above, we also retrain the model to get the counterfactual effect of skipping that stage as the gold-standard, e.g., no inserted stage or no mislabeled stage, or skipping one of the shifted stages. We then compare the estimated effects with the gold-standard results as in Table 1. We show the correlation between the estimated effects and the gold-standard results on a test set of randomly sampled MNIST data points. We see the average correlation is 0.7314, in-

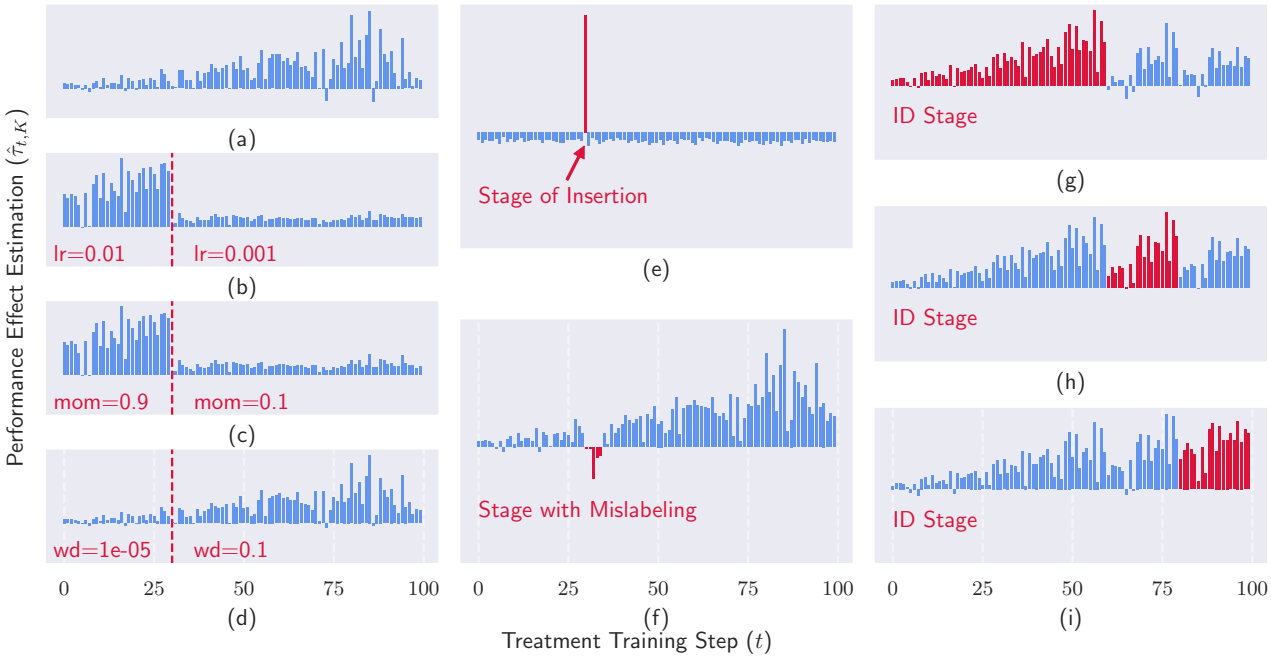

Figure 1. Performance effect of training steps on MNIST. A **positive** $\hat{\tau}_{t,K}$ (AA-Score) indicates that the training stage leads to a **higher log-likelihood**, i.e., the stage is **beneficial**. (a) Baseline case for optimization parameters. (b) Higher/lower lr leads to higher/lower performance effect. (c) Higher momentum leads to increased effect and more distributed effect across steps. (d) Stronger weight decay leads to oscillatory effect with some steps having negative effect. (e) Detect an influential stage of a data point inserted into the training process to be the inserted stage itself. (f) Capture a stage processing mislabeled data, demonstrating their negative effect on the test performance. (g-i) The stage with the highest effect on the test set is the In distribution (ID) training stage. (g) original test set. (h) 45-degree rotated test set. (i) 90-degree rotated test set.

Table 1. The correlation of AA-Score and the gold-standard results obtained by retraining the model on a test set of randomly sampled MNIST data points. The high average correlation indicates that AA-Score successfully captures the effect of the training stage on the different test data.

Setting	Insertion	Mislabel	Shift 1	Shift 2	Shift 3	Average
Corr	0.9712	0.3381	0.8430	0.7773	0.7276	0.7314

dicating that our method can capture the effect of training stages on the model's performance quantitatively. The only exception is the case of mislabeled data, where the correlation is only 0.3381, which we hypothesize is because the mislabeled data is less natural compared to the other cases, making the estimated effect less reliable.

5.3. Detect spurious correlations on CELEBA and CIVILCOMMENTS

We investigate whether our accountability attribution method can identify and mitigate spurious correlations—features that are predictive during training but not causally related to the target label. We examine two benchmark datasets with documented spurious attributes: CELEBA and CIVILCOMMENTS. In CELEBA, we study the binary classification task of predicting whether a person is *blonde*, where hair color is spuriously correlated with *gender* (Koh et al., 2021). In CIVILCOMMENTS, we analyze toxicity detection, where the *demographic identity terms* like race and religion in the comments are spuriously correlated with toxic labels (Borkan et al., 2019).

For each dataset, we designate the ground-truth label (blonde or toxicity) as the *real target* and the correlated attribute (gender or demographic identity) as the *confounding attribute*. We then:

1. Compute AA-Score for each training step on model performance with respect to the *confounding attribute*, identifying a training stage that most contribute to learning spurious correlations.

2. Select the top-k steps with strongest positive effect on the confounding attribute and strongest negative effect on the real target label.

Table 2. The model performance after retraining the model by skipping the stage with top AA-Score on the confounding attribute.

Dataset	Real (original)	Real (retrained)	Confound (original)	Confound (retrained)
CELEBA	0.9172	0.9385	0.5501	0.5187
CIVILCOMMENTS	0.6570	0.6660	0.4780	0.4690

3. Retrain the model while removing the selected steps and evaluate the retrained model on both the *real target label* and the *confounding attribute*.

We hypothesize that spurious correlations emerge during specific training stages, and removing these stages should reduce the model's reliance on confounding attributes. This should manifest as improved generalization on the real label while reducing performance on the confounding label. Our results in Table 2 confirm this hypothesis. For CELEBA, removing stages most responsible for gender correlation improves hair color classification while reducing gender prediction accuracy. Similarly for CIVILCOMMENTS, eliminating steps associated with geographic bias enhances toxicity classification while decreasing correlation with identity terms. These findings demonstrate our method's ability to both detect and mitigate the training-time origins of shortcut learning. We note that the performance change before and after retraining is not significant for language models, which is because we only tune the prediction head and keep the pre-trained model backbone fixed due to computational constraints. We hypothesize that the performance change will be more significant if the estimation is based on the entire model. We put experiment details and additional results in App. C.2.

6. Discussion

6.1. Limitations

While our framework enables general estimation of training stage effects, it has several limitations that suggest directions for future work. First, although our framework is general, the current estimators rely on a first-order Taylor approximation of the training dynamics, which may lead to reduced accuracy when higher-order effects play a significant role. Future work could extend our method to incorporate higher-order approximations or learned surrogates of the propagator to improve estimation accuracy. Second, while our estimators are more efficient than producing counterfactual situations through retraining, they remain computationally expensive for large-scale models due to the high dimensionality of propagator matrices and Hessian matrix computations, as discussed in §4.3. Future work could address this limitation by scaling the framework to foundation models through structured approximations (e.g., low-rank methods) and efficient distributed computation. Third, we have primarily conducted experiments on small to medium-sized models and datasets, using pretrained model checkpoints from the literature to study fine-tuning effects. The generalizability of our approach to pretraining-scale language models or foundation models remains an open question for future research. Finally, our framework assumes that the training pipeline and optimization history are faithfully recorded and observable. In real-world scenarios with incomplete or inaccessible training logs, applying our method may require additional assumptions or approximations, such as using stored major model checkpoints to approximate the complete training process.

6.2. Broader Impacts

Our work advances AI accountability by providing tools that trace and quantify how specific training stages influence model behavior. By localizing responsibility within the training process, these tools enhance model transparency, facilitate debugging, and enable responsible deployment. For instance, developers can identify harmful training phases that encode bias or memorize toxic data, allowing for targeted interventions and retraining. However, this framework carries potential risks. Attribution scores may be misinterpreted or misused to unfairly assign blame in collaborative model development. Malicious actors could exploit the framework to obscure training provenance or evade regulatory oversight. Like other interpretability tools, users may place excessive trust in the method's precision, particularly beyond its intended scope. We advise using accountability attribution cautiously and alongside other auditing practices. Future work should explore integrating accountability attribution into secure training pipelines to prevent misuse.

7. Conclusion

In this paper, we introduced the problem of accountability attribution, which traces model behavior to specific stages of the training process. Our key contributions include: formulating this novel accountability attribution problem; developing a general framework based on potential outcomes and counterfactual queries about training stage effects; deriving efficient estimators that account for complex optimization dynamics like learning rate schedules, momentum, and weight decay; and demonstrating practical utility by uncovering influential stages responsible for both beneficial and harmful model behaviors across diverse settings. Empirically, we showed how our framework enables attributing model behavior to training stages in a principled way. We

hope this work takes a step toward more transparent, interpretable, and accountable AI development by providing tools to analyze and assign responsibility within complex training pipelines.

References

The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Bae, J., Lin, W., Lorraine, J., and Grosse, R. B. Training data attribution via approximate unrolling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=3NaqGg92KZ.

Barshan, E., Brunet, M.-E., and Dziugaite, G. K. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pp. 1899–1909. PMLR, 2020.

Borkan, D., Dixon, L., Sorensen, J., Thain, N., and Vasserman, L. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pp. 491–500, 2019.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

Chockler, H. and Halpern, J. Y. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research*, 22:93–115, 2004.

Cole, S. R. and Frangakis, C. E. The consistency statement in causal inference: a definition or an assumption? *Epidemiology*, 20(1):3–5, 2009.

Cook, R. D. and Weisberg, S. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Ghorbani, A. and Zou, J. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Halpern, J. and Kleiman-Weiner, M. Towards formal definitions of blameworthiness, intention, and moral responsibility. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Halpern, J. Y. and Pearl, J. Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*, 2005.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Holland, P. W. Statistics and causal inference. *Journal of the American statistical Association*, 81(396):945–960, 1986.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. Datamodels: Understanding predictions with data and data with predictions. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9525–9587, 17–23 Jul 2022.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671, 2019.

Lesci, P., Meister, C., Hofmann, T., Vlachos, A., and Pimentel, T. Causal estimation of memorisation profiles. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15616–15635, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.834. URL https://aclanthology.org/2024.acl-long.834/.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

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

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Park, C. F., Okawa, M., Lee, A., Lubana, E. S., and Tanaka, H. Emergence of hidden capabilities: Exploring learning dynamics in concept space. *Advances in Neural Information Processing Systems*, 37:84698–84729, 2024.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Pearl, J. *Causality*. Cambridge university press, 2009.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Qi, Y., Schölkopf, B., and Jin, Z. Causal responsibility attribution for human-ai collaboration. *arXiv preprint arXiv:2411.03275*, 2024.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raghu, M., Zhang, C., Kleinberg, J., and Bengio, S. Transfusion: Understanding transfer learning for medical imaging. *Advances in neural information processing systems*, 32, 2019.

Ren, Y. and Sutherland, D. J. Learning dynamics of LLM finetuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tPNHOoZFl9.

Ren, Y., Guo, S., and Sutherland, D. J. Better supervisory signals by observing learning paths. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Iog0djAdbHj.

Rubin, D. B. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

Rubin, D. B. Causal inference using potential outcomes. *Journal of the American Statistical Association*, 100(469): 322–331, 2005. doi: 10.1198/016214504000001880.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Triantafyllou, S., Singla, A., and Radanovic, G. On blame attribution for accountable multi-agent sequential decision making. *Advances in Neural Information Processing Systems*, 34:15774–15786, 2021.

Wang, J. T., Song, D., Zou, J., Mittal, P., and Jia, R. Capturing the temporal dependence of training data influence. In *The Thirteenth International Conference on Learning Representations*, 2025.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Accountability Attribution Appendix

A. Derivation of Estimators 4.1 to 4.3

Here we provide the detailed steps to derive the results stated in Estimators 4.1 to 4.3.

A.1. Estimators 4.1 and 4.2

We first consider the simple case of treatment on a single step t as in Estimators 4.1 and 4.2. Recall that the treatment variable $T \in \{0, 1\}$ is defined such that $T = 0$ means step t is skipped (counterfactual), and $T = 1$ means step t is executed (observed). The causal effect of T on the final state is $\tau^{\boldsymbol{\xi}}_{K,t} = \boldsymbol{\xi}_K(1) - \boldsymbol{\xi}_K(0)$, and the causal effect on a performance function $\gamma(\boldsymbol{x}, \boldsymbol{\theta})$ is $\tau_{K,t} = Y_K(1) - Y_K(0) = \gamma(\boldsymbol{x}, \boldsymbol{\theta}_K(1)) - \gamma(\boldsymbol{x}, \boldsymbol{\theta}_K(0))$.

An interpolated path $\boldsymbol{\xi}_k(\epsilon) = (\boldsymbol{\theta}_k(\epsilon), \boldsymbol{v}_k(\epsilon))$ is defined in §4.2 such that $\boldsymbol{\xi}_k(\epsilon = 0) = \boldsymbol{\xi}_k(T = 0)$ (step t skipped) and $\boldsymbol{\xi}_k(\epsilon = 1) = \boldsymbol{\xi}_k(T = 1)$ (step t executed, observed path). We restate the interpolation here for convenience and add (observed) to indicate the observed values:

- For $k \leq t$: $\boldsymbol{\xi}_k(\epsilon) = \boldsymbol{\xi}_k(0) = \boldsymbol{\xi}_k(\text{observed})$.
- At step t: Let $\Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t+1}(\text{observed}) - \boldsymbol{\theta}_t(\text{observed})$ and $\Delta\boldsymbol{v}_t = \boldsymbol{v}_{t+1}(\text{observed}) - \boldsymbol{v}_t(\text{observed})$.

$$\boldsymbol{\theta}_{t+1}(\epsilon) = \boldsymbol{\theta}_t(\text{observed}) + \epsilon\Delta\boldsymbol{\theta}_t = \boldsymbol{\theta}_t + \epsilon(-\eta_t \boldsymbol{v}_{t+1}) \tag{13}$$

$$\boldsymbol{v}_{t+1}(\epsilon) = \boldsymbol{v}_t(\text{observed}) + \epsilon\Delta\boldsymbol{v}_t = \boldsymbol{v}_t + \epsilon(\boldsymbol{v}_{t+1} - \boldsymbol{v}_t) \tag{14}$$

- For $k > t + 1$: $\boldsymbol{\xi}_k(\epsilon)$ evolves from $\boldsymbol{\xi}_{t+1}(\epsilon)$ using standard dynamics linearized around the observed trajectory.

The first-order Taylor expansion of $\boldsymbol{\xi}_K(\epsilon)$ around $\epsilon = 1$ (the observed path) is[1]:

$$\boldsymbol{\xi}_K(\epsilon) \approx \boldsymbol{\xi}_K(1) + \frac{\partial \boldsymbol{\xi}_K(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1}(\epsilon - 1) \tag{15}$$

Get $\boldsymbol{\xi}_K(0)$ with the approximation and plug it into the effect on the state:

$$\tau^{\boldsymbol{\xi}}_{K,t} = \boldsymbol{\xi}_K(1) - \boldsymbol{\xi}_K(0) \approx \boldsymbol{\xi}_K(1) - \left(\boldsymbol{\xi}_K(1) - \frac{\partial \boldsymbol{\xi}_K(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1}\right) = \frac{\partial \boldsymbol{\xi}_K(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} \tag{16}$$

Let $\boldsymbol{w}_{K,t} = \frac{\partial \boldsymbol{\xi}_K(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1}$. Then $\tau^{\boldsymbol{\xi}}_{K,t} \approx \boldsymbol{w}_{K,t}$.

For the effect on the performance function, we similarly have

$$\tau_{K,t} \approx \frac{\partial}{\partial \epsilon}\gamma(\boldsymbol{x}, \boldsymbol{\theta}_K(\epsilon))\Big|_{\epsilon=1} \tag{17}$$

Then, apply the chain rule:

$$\tau_{K,t} \approx \nabla_{\boldsymbol{\theta}}\gamma(\boldsymbol{x}, \boldsymbol{\theta}_K(1))^\top \frac{\partial \boldsymbol{\xi}_K(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} = \nabla_{\boldsymbol{\theta}}\gamma(\boldsymbol{x}, \boldsymbol{\theta}_K)^\top [\boldsymbol{w}_{K,t}]_{\boldsymbol{\theta}} \tag{18}$$

where $[\boldsymbol{w}_{K,t}]_{\boldsymbol{\theta}}$ is the first block of the estimated difference in states, i.e., the estimated effect of difference in parameters $\boldsymbol{\theta}$.

The difference in states $\boldsymbol{w}_{K,t}$ is estimated recursively from the initial perturbation $\boldsymbol{w}_{t+1,t}$.

[1] In the literature of data attribution, e.g., influence functions (Koh & Liang, 2017), similar Taylor expansions are usually used around $\epsilon = 0$. Here we use $\epsilon = 1$ because we intend to have $\epsilon = 1$ match $T = 1$. We highlight that our expansion is equivalent to the influence function expansion, as both is around the observed outcome. The difference is that the influence function defines $\epsilon = 0$ to be the observed outcome, and it is counter-intuitive to have $T = 0$ as the observed outcome in causal inference.

Base Case (at $k = t+1$): Differentiating eqs. (13) and (14) w.r.t. ϵ (the derivative is constant):

$$\frac{\partial \boldsymbol{\theta}_{t+1}(\epsilon)}{\partial \epsilon} = -\eta_t \boldsymbol{v}_{t+1}$$
$$\frac{\partial \boldsymbol{v}_{t+1}(\epsilon)}{\partial \epsilon} = \boldsymbol{v}_{t+1} - \boldsymbol{v}_t$$

Thus, the initial perturbation $\boldsymbol{w}_{t+1,t}$ (evaluated at $\epsilon = 1$, though it's constant) is:

$$\boldsymbol{w}_{t+1,t} = \begin{pmatrix} -\eta_t \boldsymbol{v}_{t+1} \\ \boldsymbol{v}_{t+1} - \boldsymbol{v}_t \end{pmatrix} \tag{8}$$

Recursive Step (for $k > t+1$): Differentiating the SGD update rules (eqs. (1) to (4)) for the interpolated path w.r.t ϵ at $\epsilon = 1$:

$$\frac{\partial G_k(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} = H_k \frac{\partial \boldsymbol{\theta}_k(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1}$$
$$\frac{\partial {G_k}^{wd}(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} = (H_k + \lambda I) \frac{\partial \boldsymbol{\theta}_k(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1}$$
$$\frac{\partial \boldsymbol{v}_{k+1}(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} = (H_k + \lambda I) \frac{\partial \boldsymbol{\theta}_k(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} + \mu \frac{\partial \boldsymbol{v}_k(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1}$$
$$\frac{\partial \boldsymbol{\theta}_{k+1}(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} = (I - \eta_k(H_k + \lambda I)) \frac{\partial \boldsymbol{\theta}_k(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1} - \eta_k \mu \frac{\partial \boldsymbol{v}_k(\epsilon)}{\partial \epsilon}\Big|_{\epsilon=1}$$

This leads to the matrix recurrence $\boldsymbol{w}_{k+1,t} = \mathbf{M}_k \boldsymbol{w}_{k,t}$, where

$$\mathbf{M}_k = \begin{pmatrix} \mathbf{I} - \eta_k(H_k + \lambda \mathbf{I}) & -\eta_k \mu \mathbf{I} \\ H_k + \lambda \mathbf{I} & \mu \mathbf{I} \end{pmatrix} \tag{9}$$

Unrolling the recurrence:

$$\boldsymbol{w}_{K,t} = \left(\prod_{k=K-1}^{t+1} \mathbf{M}_k \right) \boldsymbol{w}_{t+1,t} \tag{19}$$

Letting $\mathbf{P}^{((t+1)\to K)} = \prod_{k=K-1}^{t+1} \mathbf{M}_k$, we have $\hat{\tau}^{(t)}_{\boldsymbol{\xi}} = \boldsymbol{w}_{K,t} = \mathbf{P}^{((t+1)\to K)} \boldsymbol{w}_{t+1,t}$. This establishes Estimator 4.1.

The estimator of the performance effect is $\tau_{K,t} \approx \nabla_{\boldsymbol{\theta}} \gamma(\boldsymbol{x}, \boldsymbol{\theta}_K)^\top [\boldsymbol{w}_{K,t}]_{\boldsymbol{\theta}}$ as in eq. (18). Let $\mathbf{P}^{((t+1)\to K)} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{pmatrix}$. The top block of $\boldsymbol{w}_{K,t}$ is:

$$\begin{aligned} [\boldsymbol{w}_{K,t}]_{\boldsymbol{\theta}} &= \mathbf{P}_{11} [\boldsymbol{w}_{t+1,t}]_{\boldsymbol{\theta}} + \mathbf{P}_{12} [\boldsymbol{w}_{t+1,t}]_{\boldsymbol{v}} \\ &= \mathbf{P}_{11}(-\eta_t \boldsymbol{v}_{t+1}) + \mathbf{P}_{12}(\boldsymbol{v}_{t+1} - \boldsymbol{v}_t) \\ &\stackrel{\text{def}}{=} E_t \end{aligned}$$

Substituting this gives the explicit form in Estimator 4.2 (eq. (11)).

A.2. Estimator 4.3

Let $S = \{t_1, \ldots, t_s\}$ be the set of distinct steps. The treatment $T_S = 1$ means all steps in S are executed, and $T_S = 0$ means all steps in S are skipped. The state effect is $\tau^{\boldsymbol{\xi}}_{K,S} = \boldsymbol{\xi}_K(T_S = 1) - \boldsymbol{\xi}_K(T_S = 0)$.

We introduce a vector of interpolation parameters $\boldsymbol{\epsilon} = (\epsilon_{t_1}, \ldots, \epsilon_{t_s})$, where $\epsilon_{t_i} \in [0, 1]$. Let $\boldsymbol{\xi}_k(\boldsymbol{\epsilon})$ denote the state on an interpolated path. For each $t_i \in S$, $\epsilon_{t_i} = 1$ means step t_i is executed, and $\epsilon_{t_i} = 0$ means step t_i is skipped. For steps $k \notin S$, the standard dynamics apply (i.e., they are executed). The state where all steps in S are executed is $\boldsymbol{\xi}_K(\mathbf{1})$. The state where all steps in S are skipped is $\boldsymbol{\xi}_K(\mathbf{0})$.

The multivariate first-order Taylor expansion of $\boldsymbol{\xi}_K(\boldsymbol{\epsilon})$ around $\boldsymbol{\epsilon} = \mathbf{1}$ is:

$$\boldsymbol{\xi}_K(\boldsymbol{\epsilon}) \approx \boldsymbol{\xi}_K(\mathbf{1}) + \sum_{t_i \in S} \frac{\partial \boldsymbol{\xi}_K(\boldsymbol{\epsilon})}{\partial \epsilon_{t_i}}\bigg|_{\boldsymbol{\epsilon}=\mathbf{1}} (\epsilon_{t_i} - 1) + o(||\boldsymbol{\epsilon}||)$$

Get $\boldsymbol{\xi}_K(\mathbf{0})$ with the approximation and plug it into the effect on the state:

$$\tau^{\boldsymbol{\xi}}_{K,S} = \boldsymbol{\xi}_K(\mathbf{1}) - \boldsymbol{\xi}_K(\mathbf{0}) \approx \sum_{t_i \in S} \frac{\partial \boldsymbol{\xi}_K(\boldsymbol{\epsilon})}{\partial \epsilon_{t_i}}\bigg|_{\boldsymbol{\epsilon}=\mathbf{1}} \tag{20}$$

The term $\frac{\partial \boldsymbol{\xi}_K(\boldsymbol{\epsilon})}{\partial \epsilon_t}\Big|_{\boldsymbol{\epsilon}=\mathbf{1}}$ is the first-order effect of step t_i, given that all other steps in S are skipped, and all steps not in S are executed. This is precisely the definition of $\hat{\tau}^{\boldsymbol{\xi}}_{K,t} = \boldsymbol{w}_{K,t}$ from Estimator 4.1 if we consider the "base" for that single-step effect to be the trajectory where t_i is skipped but all other steps (including those in $S \setminus \{t_i\}$) are executed. The linearity of the Taylor expansion allows this summation.

Thus,

$$\hat{\tau}^{\boldsymbol{\xi}}_{K,S} = \sum_{t_i \in S} \boldsymbol{w}_{K,t_i} = \sum_{t_i \in S} \hat{\tau}^{\boldsymbol{\xi}}_{K,t_i}$$

This proves the state effect in eq. (12). The proof for the performance effect $\hat{\tau}_{K,S}$ follows directly as in App. A.1.

B. Datasets and models

B.1. Dataset details

MNIST (lec): A standard benchmark dataset of handwritten digits. Due to its simplicity, it will allow for thorough case studies, including direct comparison with retraining to assess the accuracy of our approximation under various conditions (e.g., different inserted stages, different optimizers).

CELEBA (Liu et al., 2015): A large-scale face attributes dataset. This dataset is known to contain potential spurious correlations (e.g., gender with hair color). We aim to use our method to identify training stages where such spurious correlations might be predominantly learned by the model.

CIVILCOMMENTS (Borkan et al., 2019): A dataset of public comments labeled for toxicity and whether they contain words corresponding to demographic information like race, gender, and religion. This text dataset is often used for studying fairness and bias. We investigate if our method can identify training stages that disproportionately contribute to the model learning biases or relying on spurious correlations between certain identity terms and toxicity labels.

B.2. Model details

For each dataset, we employ model architectures appropriate to the task.

For MNIST, we implement a three-layer MLP architecture with 128 hidden dimensions in each layer. This relatively simple architecture allows us to perform detailed analysis of training dynamics and enables direct comparison with retraining experiments, while still providing sufficient capacity to learn meaningful digit representations.

For CELEBA, we utilize a ResNet-18 (He et al., 2016) model pre-trained on ImageNet as our backbone architecture. We augment this model with an additional final classification layer specifically trained to predict whether a celebrity has blonde hair.

For the CIVILCOMMENTS dataset, we employ a pre-trained GPT-2 model (Radford et al., 2019) from the Huggingface library (Wolf et al., 2019) as our base architecture. We extend this model with a classification head trained to predict comment toxicity.

The training hyperparameters are different for each experiment, which we specify in the following sections.

C. Detailed experiment settings and additional results

C.1. Accountability attribution on MNIST

C.1.1. Effect of optimization parameters

For this experiment, we train a 2-layer MLP with hidden dimension 128 on a subset of MNIST of the first 10,000 samples for 1 epoch with batch size 100 and learning rate 0.01. We investigate three key parameters: learning rate (lr), momentum (mom), and weight decay (wd). When studying each parameter, we keep the others constant across stages.

First, we show the baseline case of one stage, all three parameters stay the same, with lr=0.01, momentum=0.9, and wd=1e-5. These are the common settings for training MLPs on MNIST and the parameters for the first stage for all other settings. As shown in Fig. 1 (a), the effects are getting larger as the training goes on, which is expected as the model will forget earlier stages so the effect of the later stages will be larger.

For learning rate experiments, we train with lr=0.01 in stage 1 and lr=0.001 in stage 2. As shown in Fig. 1 (b), our method captures how lower learning rates lead to decreased stage effects, matching the intuition that smaller updates have less impact on model parameters.

For momentum experiments, we set stage 1 momentum to 0.9 and vary stage 2's momentum to 0.1. Fig. 1 (c) shows that lower momentum leads to decreased effect magnitude similar to the learning rate experiment, and broader distribution of effects across steps, reflecting how momentum accumulates and propagates update impacts from earlier stages.

For weight decay experiments, we use wd=1e-5 in stage 1 and vary stage 2's wd to 0.01. In Fig. 1 (d), we observe that larger weight decay leads to scores in the second stage become smaller in magnitude, for both positive and negative scores. This is because the meaningful learning signals come from the data are less significant with larger weight decay.

We also show additional experiments with different parameter settings in Fig. 2.

C.1.2. Detect an influential training stage

For this experiment, we train a 2-layer MLP with hidden dimension 128 on a small subset of MNIST of the first 100 samples, excluding digit '4'. We train for 1 epoch with batch size 1 using a learning rate of 0.001. We use a small subset and batch size 1 so we can insert a single instance of digit '4' at any training step. At training step 30, we insert a single instance of digit '4' from the test set to study its effect on itself, other images of digit '4', and other digits.

In Fig. 1 (e), we show the effect of the training stages estimated by AA-Score. We can see that our attribution scores correctly identify the inserted step as having the highest positive effect on the model's performance on the same digit '4' classification. In Fig. 3, we further analyze the effect of inserting a test digit '4' during training on the model's ability to classify other digits, with Fig. 3 (a) being the same case as Fig. 1 (e) for effect on the same digit '4' that is inserted. For the other three plots, we pick another digit '4' from the test set different from the one inserted in Fig. 3 (b), a digit '9' which is easily confusable as the digit '4' in Fig. 3 (c), and a digit '2' which is visually distinct from '4' in Fig. 3 (d). We observe that the inserted step has the strongest positive effect on classifying other digit '4's, showing that the model learns generalizable features. The effect is slightly negative for digit '9', which shares some visual features with '4', suggesting learning the inserted digit '4' has negative effect on digit '9's classification. For digit '2', which is visually distinct from '4', the effect is close to neutral, slightly negative but not as large as the effect on '9', indicating that the learning is specific to relevant digit features.

C.1.3. Capture a negative stage caused by mislabeled data

For this experiment, we train a 2-layer MLP with hidden dimension 128 on a subset of MNIST of the first 10,000 samples for 1 epoch with batch size 100 and learning rate 0.01. We introduce label noise by flipping labels for 5% of the training samples. Specifically, starting from the 30th step, we modify labels of five consecutive batches (500 samples total) through a cyclic shift (digit $0\rightarrow1$, $1\rightarrow2$, etc.).

In Fig. 1 (f), we analyze the effect of these mislabeled training stages. Our method successfully identifies these stages as having significant negative effects on the model's test performance. The magnitude of negative effects correlates with the degree of label shift. This demonstrates our method's ability to quantify the harmful impact of noisy training data.

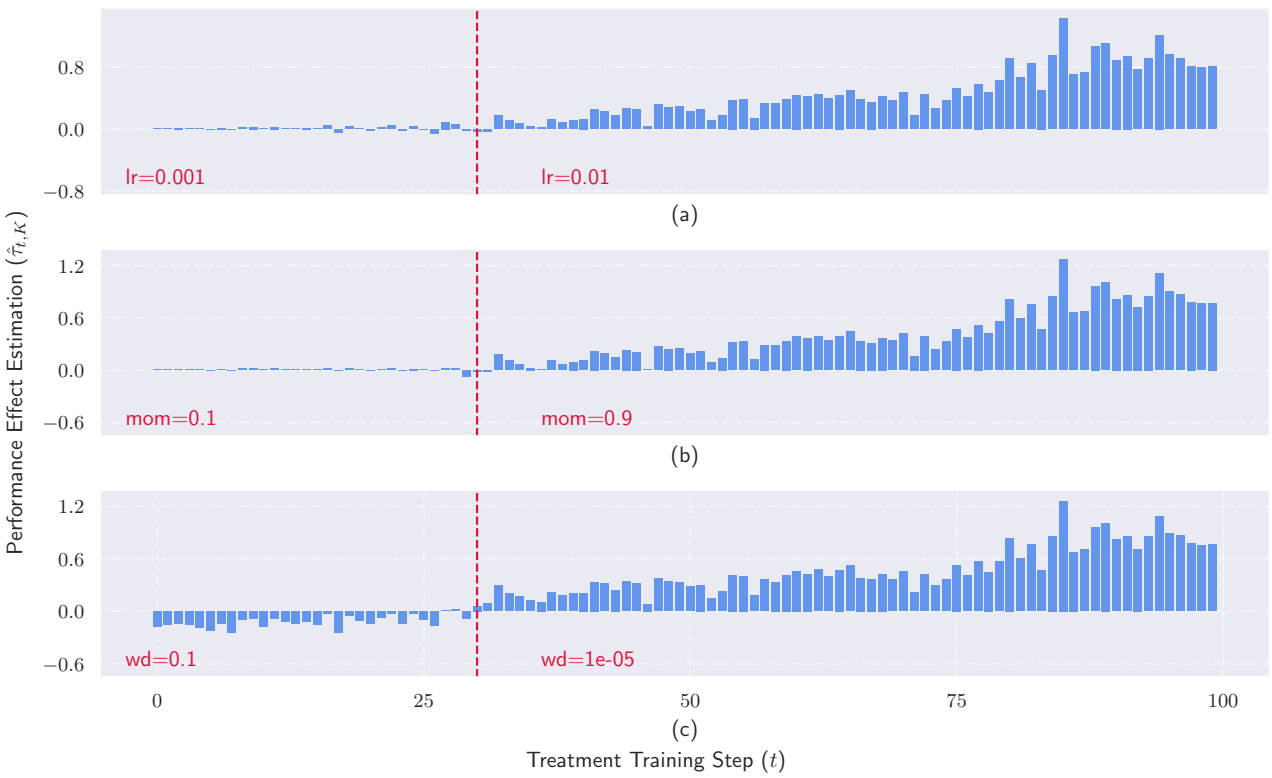

Figure 2. Additional experiments on the effect of different parameters on the model's performance. (a) is the effect of different learning rates. (b) is the effect of different momentums. (c) is the effect of different weight decays.

C.1.4. Multi-stage training with distributional shifts

For this experiment, we train a 2-layer MLP with hidden dimension 128 on a subset of MNIST of the first 2,000 samples with batch size 100 and learning rate 0.01. We introduce a distributional shift by rotating the images by 45 degrees in stage 2, and by 90 degrees in stage 3. We train for 3 epochs for stage 1, 1 epoch for stage 2, and 1 epoch for stage 3.

In Fig. 1 (g-i), we evaluate each stage's effect on three test sets: original orientation, 45-degree rotated, and 90-degree rotated. The results show clear specialization - each stage has maximum effect on its corresponding test distribution. For example, stage 2 (45-degree training) shows the highest positive effect on 45-degree rotated test images.

We also observe some transfer effects between stages. Training on 45-degree rotated images (stage 2) shows moderate positive effects on both 0-degree and 90-degree test sets, suggesting the model learns some rotation-invariant features. However, the 90-degree stage shows minimal positive effect on 0-degree test performance, indicating potential catastrophic forgetting of the original orientation when the distributional shift is too large.

C.2. Accountability attribution on CelebA and CivilComments

C.2.1. Experiment details and additional results on CelebA

For CelebA experiments, we train the model on a subset of 1628 images from the dataset for 10 epochs with batch size 1 and learning rate 0.0003, momentum 0.9, and weight decay 0.00001. To investigate potential spurious correlations, we simultaneously evaluate the model's implicit learning of gender information. This dual evaluation setup allows us to assess whether the model truly learns to classify hair color or if it relies on gender as a confounding variable in its decision-making process.

For evaluation, we partition the test set into four demographic categories: *blonde-haired males*, *non-blonde-haired males*,

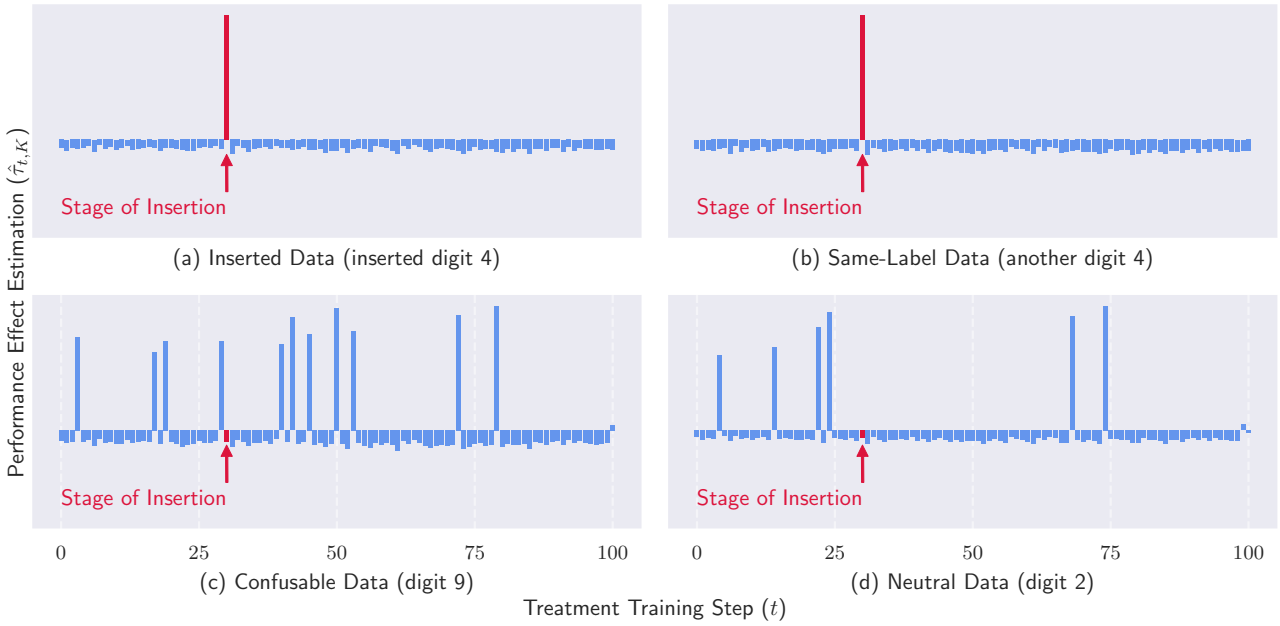

Figure 3. The effect of inserting a test digit '4' during training on the model's ability to classify four different digits. (a) is the same case as Fig. 1 (e) for effect on the same digit '4' that is inserted. (b) is the effect on another digit '4' from the test set. (c) is the effect on digit '9', which is easily confusable as '4'. (d) is the effect on a neutral digit '2', which is visually distinct from '4'.

blonde-haired females, and *non-blonde-haired females*. We compute the average AA-Score for each category to analyze learning dynamics across groups and epochs. The results show that AA-Score effectively capture variations in learning trajectories, particularly distinguishing patterns between blonde and non-blonde groups. As shown in Fig. 4, the final epoch yields the highest AA-Score, whereas mid-training epochs produce the lowest. The initial stage yields slightly higher magnitude of effects than the middle stages. We hypothesize that this happens due to continue learning catastrophic forgetting for early stages, the initial stage has more effect because they correspond to the initial representation learning. When we observe the test loss for each training epoch in Fig. 5, in the 10^{th} epoch, both Male Blonde and Female Blonde categories exhibit a clear performance drop, while loss increases for non-blonde groups. These trends are reflected in the AA-Score: the sign of the score indicates a positive effect for blonde categories and a negative impact for non-blonde categories.

Our method also surfaces mislabeled training examples. In the CELEBA dataset, we discovered a striking case: a data point corresponding to a *blonde-haired man* that was incorrectly labeled as *not blonde*. Our accountability attribution framework identified this instance as having the *most negative contribution* to the model's prediction performance on the true blonde label—it was ranked at the bottom when sorted by causal effect on the target performance. Upon inspection, we verified that the image was indeed mislabeled. This example highlights the diagnostic capability of our method: by tracing the impact of individual training steps or data points, it can surface outliers or label noise that would be difficult to detect through aggregate metrics alone.

C.2.2. EXPERIMENT DETAILS ON CIVILCOMMENTS

For CIVILCOMMENTS experiments, we train the model on a subset of 2000 comments from the dataset for 5 epochs with batch size 100 and learning rate 0.00001, momentum 0.9, and weight decay 0.00001. We also set the maximum gradient norm to 10.0 to stabilize the training. To investigate potential biases, we specifically analyze the model's behavior regarding the confounding variable, e.g., the identity terms "Christian" in the comments. This allows us to evaluate whether the model genuinely learns to classify toxicity or if it develops undesirable associations with specific demographic identifiers.

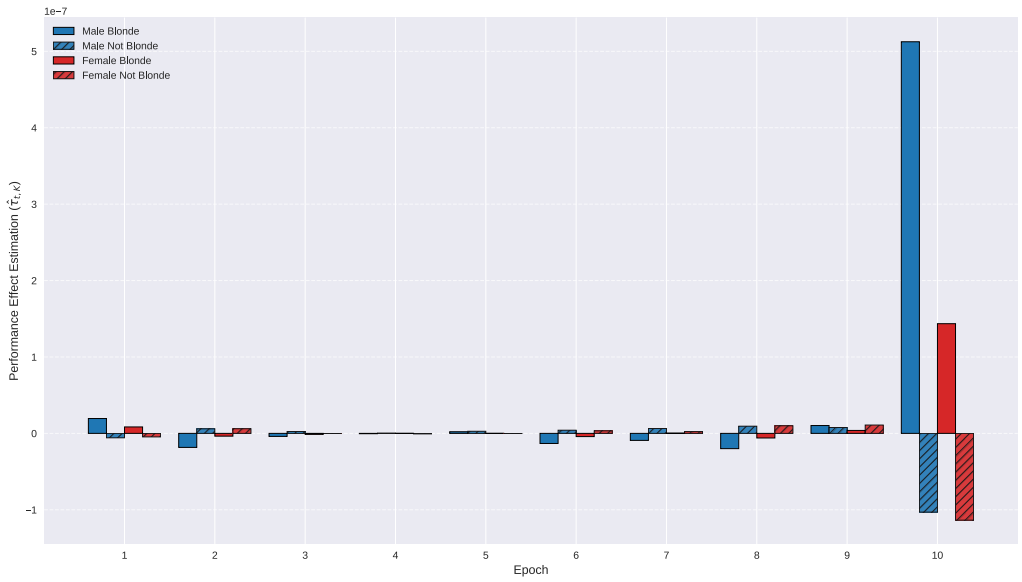

Figure 4. Effect of different training stages on the model's ability to classify hair color (grouped by four demographic categories within each epoch).

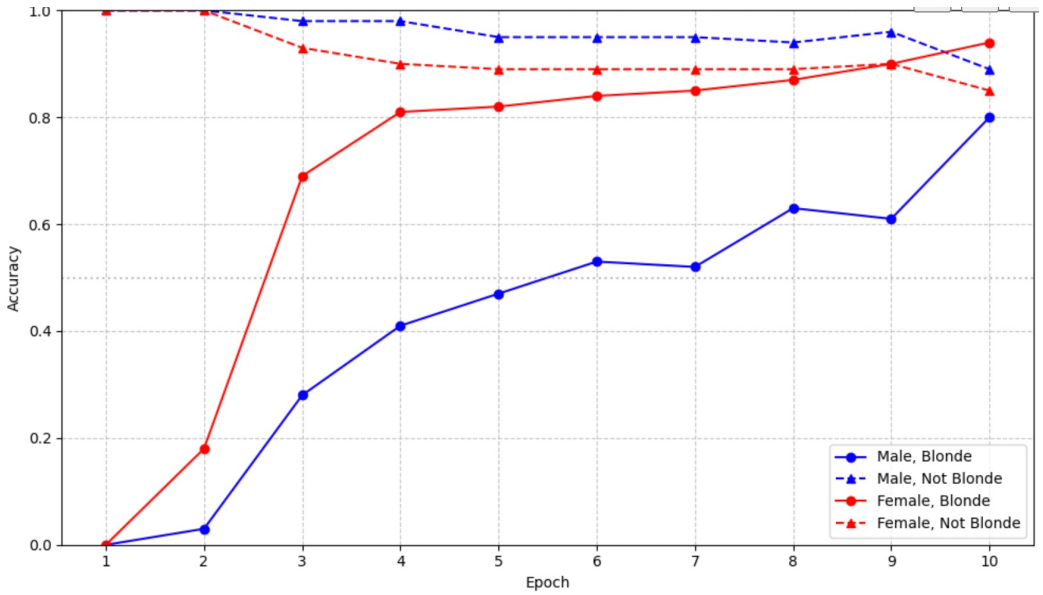

Figure 5. Test accuracy of the model for each training epoch.

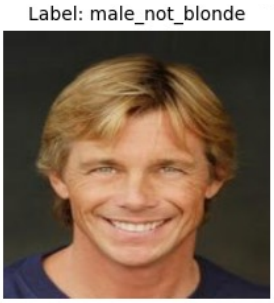

Figure 6. A mislabeled training example that is identified by our method as having the most negative contribution to the model's prediction performance on the true blonde label.