DISTILLING DIFFUSION SCORE DISCREPANCY FOR EFFICIENT TRAINING DATA ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Training data attribution for diffusion models aims to identify the training samples that influence a generated instance, but existing methods either require costly per-sample gradient computation or query-specific model optimization. Moreover, most methods attribute changes in a proxy loss rather than changes in the actual model's generative behavior. We address these limitations by formulating attribution directly with a *local score discrepancy* measure, which applies to any diffusion variant (including DDPM, EDM, and flow matching), and by showing that such measure can be estimated without retraining, as a preconditioned gradient similarity. We instantiate this estimator as ***T**raining-data **I**nfluence via score **D**iscrepancy* (TID), which uses Kronecker-factored curvature to avoid random projections and per-sample gradient storage. We then distill TID into TIDE, a forward-only student trained online to reproduce the teacher's rankings from the diffusion model's internal activations. Under counterfactual evaluation on CIFAR-10, ArtBench-10, and MS-COCO, TID matches or outperforms state-of-the-art approaches, while TIDE retains most of TID's accuracy at four to five orders of magnitude lower per-query cost, attributing generated samples in milliseconds and faster than the generation itself.

1 INTRODUCTION

Diffusion models are trained on large-scale collections of human-created images and have demonstrated remarkable capabilities in synthesizing diverse, high-quality visual content (Ho et al., 2020; Song et al., 2021a; Karras et al., 2022; Lipman et al., 2023). Their widespread adoption, however, has also raised growing concerns regarding copyright infringement (Zhang et al., 2024; Lu et al., 2024), memorization (Somepalli et al., 2023; Carlini et al., 2023), and other data-related issues. These concerns have motivated increasing interest in training data attribution (TDA) for diffusion models (Dai & Gifford, 2023; Georgiev et al., 2023; Zheng et al., 2024; Wang et al., 2024; Lin et al., 2025; Serrà et al., 2026), which aims to identify the training samples that most strongly influence a particular generated image.

Despite the growing body of work on diffusion TDA, two main issues remain open. The first concerns the formulation: most gradient-based estimators measure the counterfactual effect of a training sample on a proxy utility, typically the training loss (Georgiev et al., 2023; Zheng et al., 2024), whereas the question asked of a generative model is how its training data determines what it generates. A shift in loss does not necessarily reflect a shift in the generated output (Lin et al., 2025). The second concerns the cost: gradient-based attribution (Georgiev et al., 2023; Zheng et al., 2024; Lin et al., 2025) requires per-sample gradients for both training and query samples across multiple diffusion timesteps, at a cost that can approach that of model training. Unlearning-based attribution (Ko et al., 2024; Wang et al., 2024; Serrà et al., 2026) reduces the training side to forward passes only, but requires a dedicated fine-tuning procedure for each query, so the overall cost still does not scale. Appendix A provides a detailed review of existing approaches.

In this work, we address both aforementioned issues. For the formulation issue, we start from the observation that sampling in a diffusion model is driven by its learned score (Song et al., 2021b; Song & Ermon, 2020), so removing a training sample can change what is generated for a query mainly by perturbing the score along the query's forward noising. We therefore define the influence of a training sample as the expected *score discrepancy* between the original and the perturbed model *at*

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

different noisy states of the query. This target is generalizable for most diffusion variants, including DDPM (Ho et al., 2020), EDM (Karras et al., 2022), and flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023), and a first-order expansion turns it into a preconditioned gradient similarity that can be evaluated without retraining. We instantiate this estimator as ***T**raining-data **I**nfluence via score **D**iscrepancy* (TID). To avoid the distortion introduced by random projections, TID preconditions gradients with layer-wise Kronecker-factored curvature (Martens & Grosse, 2015; George et al., 2018), which also removes the need for per-sample gradient storage.

For the cost issue, we follow the recent idea of influence distillation (Wang et al., 2025) and distill TID into a forward-only student model built directly from the diffusion model's internal representations, eliminating the need for a separate visual encoder. We term this model ***TID E**mbeddings* (TIDE). Rather than constructing a fixed distillation dataset offline (Wang et al., 2025), our TID formulation enables the generation of teacher supervision online within each training batch. This strategy results in efficient post hoc training while exposing the student to a broader and continually evolving set of attribution targets. TIDE produces compact, low-dimensional embeddings that support efficient nearest-neighbor retrieval under standard similarity metrics, the core operation required to attribute a generated instance.

Our main contributions are as follows:

- We formulate diffusion TDA as the expected local score discrepancy induced by removing a training sample from a query's forward noising, a target that is expressed directly in the model's generative mechanism and applies across multiple diffusion variants.
- We instantiate this target as TID (Sec. 2.1), a gradient-based teacher that uses Kronecker-factored curvature to avoid both random projections and per-sample gradient storage.
- We distill TID into TIDE (Sec. 2.2), a forward-only student trained online from the diffusion model's own activations using a top-heavy ranking objective.
- Counterfactual retraining on CIFAR-10, ArtBench-10, and MS-COCO shows that TID outperforms state-of-the-art unlearning-based TDA on two datasets, while TIDE retains most of its accuracy at four-to-five orders of magnitude lower per-query cost. Ablations validate our core design choices (Secs. 3.2–3.4).

2 METHOD

In this section, we introduce ***T**raining-data **I**nfluence via score **D**iscrepancy* (TID, Sec. 2.1), a general gradient-based attribution method applicable to most diffusion-model variants. We define influence through a local diffusion score discrepancy and derive an efficiently computable gradient similarity score to estimate it. Next, we show that this score can be distilled into the diffusion model's own forward representations, yielding ***TID E**mbeddings* (TIDE, Sec. 2.2), further removing the remaining backward passes during attribution and compacting the embeddings into a low-dimensionality space. Additional overview and a block diagram can be found in Appendix B.

Notation. Consider a general diffusion model under common principles (Lai et al., 2025), parametrized by θ, with a learnt score s_θ obtained from a neural network F_θ. The latter is pre-trained with a reconstruction loss L that involves noise scales σ, noise vectors $\mathbf{n}$, and training instances $\mathbf{z}$. Each instance consists of a data item $\mathbf{x}$ and, optionally, a conditioning signal $\mathbf{c}$, i.e., $\mathbf{z} = \{\mathbf{x}, \mathbf{c}\}$. We denote the training set by $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{|\mathcal{Z}|}\}$. At inference time, the model generates an item $\hat{\mathbf{x}}$ by iterative denoising along a decreasing schedule of noise scales $\sigma_{\max} = \sigma_T > \cdots > \sigma_1 = \sigma_{\min}$ using conditioning $\hat{\mathbf{c}}$. We write $\hat{\mathbf{z}} = \{\hat{\mathbf{x}}, \hat{\mathbf{c}}\}$ for the resulting attribution query. We also use $\mathbf{x}_{\sigma,\mathbf{n}}$ and $\hat{\mathbf{x}}_{\sigma,\mathbf{n}}$ to denote the forward noising of a training or a generated item, respectively (see Appendix C.1).

2.1 LOCAL SCORE DISCREPANCY

TDA asks how removing a training sample $\mathbf{z}_i$ changes a model's behavior at a given query $\hat{\mathbf{z}}$. For a generative model, this effect is naturally characterized by how the generation produced by the leave-one-out model $F_{\theta^{\setminus i}}$ differs from the original generation, where $\theta^{\setminus i}$ denotes the parameters obtained by retraining from scratch without $\mathbf{z}_i$. To measure this effect, existing work usually relies on indirect counterfactual proxies, such as changes in the training loss (Georgiev et al., 2023; Zheng

et al., 2024; Mlodozeniec et al., 2025). In a diffusion model, however, generation is driven by the learned score field (Song et al., 2021b; Song & Ermon, 2020). Perturbations to the training distribution induce changes in the optimal score field, which propagate through the sampling dynamics to affect generated samples (Scarvelis & Solomon, 2025). We therefore quantify the removal effect in this space, comparing the score functions of s_θ and $s_{\theta^{\setminus i}}$ along noisy states originating from the fixed query/generated item. To do so, we consider a Monte Carlo estimate using a sample of noise scales σ and noise realizations $\mathbf{n}$, and study the score discrepancy at a given anchor point $\hat{\mathbf{z}}$ (the query/generated item). We define the *score discrepancy at* $\hat{\mathbf{z}}$ caused by $\mathbf{z}_i$ as

$$\tau_\theta(\mathbf{z}_i; \hat{\mathbf{z}}) := \mathbb{E}_{\sigma,\mathbf{n}}\left[\left\| s_\theta\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right) - s_{\theta^{\setminus i}}\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right)\right\|_2^2\right],$$

where s_θ and $s_{\theta^{\setminus i}}$ denote the scores learned by the models using $\mathcal{Z}$ and $\mathcal{Z} \setminus \mathbf{z}_i$, respectively. Intuitively, a larger $\tau_\theta(\mathbf{z}_i; \hat{\mathbf{z}})$ indicates that removing $\mathbf{z}_i$ shifts the learned score field more strongly around the query $\hat{\mathbf{z}}$.

Because the score estimate is common to most diffusion variants, this definition is agnostic to any particular parameterization: it applies whenever the generative model provides a noise- or a time-indexed score, including DDPM (Ho et al., 2020), EDM (Karras et al., 2022), and flow-matching models (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023). Moreover, for the common parameterizations, the score estimate is an affine function of the raw network output with schedule-dependent coefficients (Lai et al., 2025), so the discrepancy can be evaluated on model outputs directly:

$$\tau_\theta(\mathbf{z}_i; \hat{\mathbf{z}}) = \mathbb{E}_{\sigma,\mathbf{n}}\left[w_\sigma \left\| F_\theta\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right) - F_{\theta^{\setminus i}}\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right)\right\|_2^2\right], \tag{1}$$

where the weight w_σ depends only on the noise scale σ and fully encodes the diffusion variant. Appendix C.2 derives Eq. 1 and instantiates w_σ for DDPM, EDM, and flow matching.

Linearization: from output differences to gradient similarity. Directly evaluating Eq. 1 requires retraining a new leave-one-out model $F_{\theta^{\setminus i}}$ for every training sample. Since this is computationally infeasible, we next introduce a practical instantiation tailored at more realistic deployments. We start by considering a first-order Taylor expansion around the trained parameters θ:

$$F_{\theta^{\setminus i}}\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right) \approx F_\theta\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right) + \nabla_\theta F_\theta\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right)\Delta\theta_i, \qquad \Delta\theta_i := \theta^{\setminus i} - \theta,$$

where $\nabla_\theta F_\theta\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right) \in \mathbb{R}^{D\times P}$ is the Jacobian of the model output with respect to the parameters, D is the output dimension, and $\Delta\theta_i \in \mathbb{R}^P$ is the parameter change induced by removing $\mathbf{z}_i$. Substituting into Eq. 1 gives

$$\tau_\theta(\mathbf{z}_i; \hat{\mathbf{z}}) \approx \mathbb{E}_{\sigma,\mathbf{n}}\left[w_\sigma \left\| \nabla_\theta F_\theta\left(\hat{\mathbf{x}}_{\sigma,\mathbf{n}}; \sigma, \hat{\mathbf{c}}\right)\Delta\theta_i\right\|_2^2\right]. \tag{2}$$

Since the model output is high-dimensional, we follow TRAK (Park et al., 2023) and DAS (Lin et al., 2025) to approximate its change through a scalar regression model whose output is the sum of the model output coordinates, and then estimate the counterfactual parameter change $\Delta\theta_i$ by a one-step Newton update on the removed sample. We consider the scalar regression model producing the summed output $\langle \mathbf{1}, F_\theta\rangle$ and, with such output, define the model's parameter gradient as

$$g(\mathbf{z}; \sigma, \mathbf{n}) := \nabla_\theta\left\langle \mathbf{1}, F_\theta(\mathbf{x}_{\sigma,\mathbf{n}}; \sigma, \mathbf{c})\right\rangle \in \mathbb{R}^P. \tag{3}$$

For simplicity, we will omit $(\sigma, \mathbf{n})$ from g and related quantities when there is no ambiguity. Next, to estimate the parameter change $\Delta\theta_i$, we apply the Newton's method (Pregibon, 1981) to the scalar regression model, where a one-step Newton update approximation gives the closed form

$$\Delta\theta_i \approx -\frac{r_i}{1-h_i}\,\mathbf{G}^{-1} g(\mathbf{z}_i), \qquad \mathbf{G} = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}\left[g(\mathbf{z}; \sigma, \mathbf{n})\, g(\mathbf{z}; \sigma, \mathbf{n})^\top\right], \tag{4}$$

where $\mathbf{G}$ is the curvature of the regression model, the leverage $h_i = g(\mathbf{z}_i)^\top \mathbf{G}^{-1} g(\mathbf{z}_i) \in [0, 1)$, and the training residual $r_i = \langle \mathbf{1},\ F_\theta(\mathbf{x}_{\sigma,\mathbf{n}}; \sigma, \mathbf{c}) - \mathbf{y}\rangle$, evaluated at $\mathbf{z}_i$, with $\mathbf{y}$ representing the diffusion variant's regression target (for example, $\mathbf{y} = \mathbf{n}$ for DDPM and $\mathbf{y} = \mathbf{x}$ for EDM, cf. Park et al., 2023; Lin et al., 2025). For ease of calculation, we drop the factor $r_i/(1-h_i)$, as our ablations confirm that such terms are negligible to the ranking (Sec. 3.4). Dropping such factor and substituting Eqs. 3 and 4 into Eq. 2 yields

$$\tau_\theta(\mathbf{z}_i, \hat{\mathbf{z}}) \approx \mathbb{E}_{\sigma,\mathbf{n}}\left[\, w_\sigma \left\langle g(\hat{\mathbf{z}}; \sigma, \mathbf{n}), \mathbf{G}^{-1} g(\mathbf{z}_i; \sigma, \mathbf{n})\right\rangle^2\right]. \tag{5}$$

Aligned draws and inference. In practice, the expectation in Eq. 5 is estimated by Monte Carlo sampling, with σ drawn from the diffusion sampler's own schedule. Although the Newton update imposes no restriction on the choice of noise scale σ or noise realization $\mathbf{n}$, Serrà et al. (2026) observed that aligning diffusion scheduler timesteps can dramatically improve attribution performance. Hence, to make the scores $\tau_\theta(\mathbf{z}_i; \hat{\mathbf{z}})$ directly comparable across training samples, we follow such strategy and use the same set of noise scales σ and noise realizations $\mathbf{n}$ for all the queries, with each draw $(\sigma, \mathbf{n})$ being shared among queries on both sides of the kernel in Eq. 5. Specifically, we use M noise scales of the sampler's Karras grid, each paired with a single noise vector drawn once under a fixed seed and shared across all samples and queries. (we study how M trades off cost and accuracy in Sec. 3.3).

Layer-wise kernel estimation with Kronecker factorization. Evaluating Eq. 5 requires computing the inner product kernels $\langle g(\hat{\mathbf{z}}), \mathbf{G}^{-1} g(\mathbf{z}_i) \rangle$ for all pairs. Most existing works compress those vectors with random projections, introducing projection-induced distortion on kernel estimation (Park et al., 2023; Zheng et al., 2024; Lin et al., 2025). Following K-FAC Influence (Mlodozeniec et al., 2025), we instead approximate $\mathbf{G}$ blockwise, per layer, with Kronecker-factored approximate curvature (K-FAC; Martens & Grosse, 2015; George et al., 2018). For a linear layer l with input activations $\mathbf{a}_l$ and pre-activation output gradients $\mathbf{b}_l$, the layer gradient can be decomposed as $g_l(\mathbf{z}) = \mathbf{b}_l \, \mathbf{a}_l^\top$, which gives the following Kronecker factorization:

$$\mathbf{G}_l \approx \mathbf{A}_l \otimes \mathbf{B}_l \quad \Longrightarrow \quad \left\langle g_l(\hat{\mathbf{z}}), \mathbf{G}_l^{-1} g_l(\mathbf{z}_i) \right\rangle \approx \left(\hat{\mathbf{a}}_l^\top \mathbf{A}_l^{-1} \mathbf{a}_{l,i}\right) \left(\hat{\mathbf{b}}_l^\top \mathbf{B}_l^{-1} \mathbf{b}_{l,i}\right), \tag{6}$$

where $\otimes$ is the Kronecker product, $\mathbf{A}_l = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}[\mathbf{a}_l \mathbf{a}_l^\top]$, $\mathbf{B}_l = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}[\mathbf{b}_l \mathbf{b}_l^\top]$, and $(\hat{\mathbf{a}}_l, \hat{\mathbf{b}}_l)$ and $(\mathbf{a}_{l,i}, \mathbf{b}_{l,i})$ denote the factors of the query and of $\mathbf{z}_i$, respectively. A similar factorization applies to other common layer types in diffusion models, including convolutional or transposed convolutional layers (we provide the corresponding derivations in Appendix D).

Normalization. In our implementation, per-sample and per-layer gradient factors are L2-normalized before accumulation, which is a common practice in gradient-similarity attribution that prevents samples with large gradient norms from dominating the scores (Barshan et al., 2020; Hanawa et al., 2021; Xia et al., 2024). Without normalization, gradients evaluated at larger σ, where the inputs contain more noise, can dominate the aggregation across noise levels and make the resulting rankings sensitive to the sampled noise scales. For a similar reason, we drop the weight w_σ: after normalization, each sampled noise scale contributes a standardized influence value with a common scale, and reintroducing a schedule-dependent multiplier would disturb this standardization. Putting everything together, our final attribution score becomes

$$\tau_\theta(\mathbf{z}_i; \hat{\mathbf{z}}) \approx \mathbb{E}_{\sigma,\mathbf{n}} \left[\left(\sum_l \left(\hat{\underline{\mathbf{a}}}_l^\top \underline{\mathbf{A}}_l^{-1} \underline{\mathbf{a}}_{l,i}\right) \left(\hat{\underline{\mathbf{b}}}_l^\top \underline{\mathbf{B}}_l^{-1} \underline{\mathbf{b}}_{l,i}\right) \right)^2 \right], \tag{7}$$

where $\underline{\mathbf{A}}_l = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}[\underline{\mathbf{a}}_l \underline{\mathbf{a}}_l^\top]$, $\underline{\mathbf{B}}_l = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}[\underline{\mathbf{b}}_l \underline{\mathbf{b}}_l^\top]$, and the underline indicates L2 normalization.

Relation to existing gradient-based methods. TID belongs to the broad family of gradient-based attribution methods, which includes TRAK (Park et al., 2023), D-TRAK (Zheng et al., 2024), DAS (Lin et al., 2025), and K-FAC Influence (Mlodozeniec et al., 2025). TID differs mainly in the *attribution target* being differentiated and the *curvature model* used to precondition its gradient. Furthermore, it offers a more general attribution framework. Note that, for a DDPM noise predictor, Eq. 1 reduces to the expected squared difference of noise predictions that motivates DAS. However, DAS obtains it from a KL divergence specialized to the DDPM reverse process and the loss $\mathcal{L}_{\text{simple}}$, whereas TID obtains it from the score discrepancy, which does not rely on the probabilistic parameterization of DDPM nor the specific loss being used, and hence extends to EDM and flow matching. Beyond this formulation difference, the methods differ in five design choices: (i) *attribution target*: TID differentiates the summed model output, whereas DAS and K-FAC Influence differentiate the denoising loss and D-TRAK differentiates the squared output norm; (ii) *corruption sampling*: TID evaluates query and candidates at the same $(\sigma, \mathbf{n})$, with σ drawn from the inference schedule, whereas D-TRAK, DAS and K-FAC Influence corrupt the samples independently; (iii) *gradient normalization*: TID normalizes the two Kronecker factors of each layer before aggregating the layerwise kernels, whereas DAS normalizes the whole projected gradient at each timestep and D-TRAK and K-FAC Influence use unnormalized gradients; (iv) *Newton corrections*: TID omits the residual and leverage terms r_i and $1 - h_i$ of Eq. 4, whereas D-TRAK retains a residual-type factor

$\mathcal{Q}$ and DAS retains both; and (v) *curvature estimation*: TID forms empirical-Fisher K-FAC factors from observed activations and output gradients, whereas D-TRAK and DAS use a ridge-damped Gauss–Newton kernel of randomly projected gradients, which introduces projection-induced distortion, while K-FAC Influence estimates a Monte Carlo generalized Gauss–Newton matrix by resampling the regression target. Sec. 3.4 ablates i–v and Appendix A gives a broader literature review, including non-gradient-based attribution methods.

2.2 Distilling Attribution into Forward Embeddings

While TID avoids per-sample gradient storage, each attribution score still requires a forward–backward pass per compared sample and implies a potentially high dimensionality (number of model parameters). To further reduce such costs, we additionally propose ***TID Embeddings*** (TIDE), which eliminates the need for backward passes entirely and projects to a lower dimensionality d. In this setting, TIDE becomes a student method distilled from its teacher, TID (hereafter referred to as the *teacher*). It learns an embedding $\mathbf{e} \in \mathbb{R}^d$ from the diffusion model's internal activations, obtained from a single forward pass, such that the embedding similarity reproduces the TID's attribution ranking at dimensions as low as $d = 768$; orders of magnitude below the gradient representations they replace. In principle, any attribution method could serve as the teacher, but TID's factorized, per-layer scores make distillation particularly efficient.

Embedding model architecture. TIDE uses no external encoder and its input consists of internal representations already computed by the diffusion model F_θ. Let $\mathbf{h}_k(\mathbf{z}; \sigma, \mathbf{n})$ denote the token-wise residual-stream activations after the k-th of the K transformer blocks of F_θ when processing the noised input $\mathbf{x}_{\sigma,\mathbf{n}}$. An embedding $\mathbf{e}_i$ is computed from the blockwise activations sum through a learned attention pooling over tokens, followed by a small MLP (all in a single forward pass):

$$\mathbf{e}_i = E_\phi(\mathbf{z}_i; \sigma, \mathbf{n}) = \mathrm{MLP}\left(\mathrm{AttnPool}\left(\sum_{k=1}^{K} \mathbf{h}_k(\mathbf{z}_i; \sigma, \mathbf{n})\right)\right). \tag{8}$$

The score between two embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ is their cosine similarity $\rho_{ij} = \rho_\phi(\mathbf{z}_i, \mathbf{z}_j; \sigma, \mathbf{n}) := \mathbf{e}_i^\top \mathbf{e}_j / (\|\mathbf{e}_i\| \, \|\mathbf{e}_j\|)$.

Batched pair-wise score distillation. The prior distilled attribution of Wang et al. (2025) learns its student from an unlearning-based teacher. Hence, annotating the influence of a training sample requires a dedicated unlearning run per training sample, making it substantially more expensive to expand supervision coverage on the training side than to add query samples. The supervision set is therefore curated offline for a fixed collection of pairs, and limited to the coverage budgeted before training. TID removes this asymmetry, as it requires only one single forward–backward pass per training or query sample. For each training data batch $\mathcal{B} = \{\mathbf{z}_1, \ldots, \mathbf{z}_B\}$, we draw a single shared $(\sigma, \mathbf{n})$ under the aligned regime of Sec. 2.1 and compute, in one forward–backward pass, both the within-batch pairwise TIDE scores ρ_{ij} and the teacher's within-batch pairwise scores $\tau_{ij} := \tau_\theta(\mathbf{z}_i; \mathbf{z}_j)$. Compared to offline annotation, this has two advantages: (i) *coverage*: every step supervises a set of B^2 pairs, so the supervised pairs accumulate throughout training rather than being fixed in advance; and (ii) *no annotation overhead*: supervision is computed when it is consumed and is never materialized or stored. We provide the distillation algorithm summary in Appendix B.

Ranking objective. Attribution is usually consumed as per-query rankings (Ghorbani & Zou, 2019; Xia et al., 2024; Wang et al., 2025), so we distill rankings rather than raw score values. Pointwise regression of scores or normalized ranks, as in Wang et al. (2025), weights every candidate equally and under-constrains the top of the ranking. In contrast, we adopt a top-heavy pairwise objective from the LambdaLoss family (Burges et al., 2005; Wang et al., 2018). Within a batch, we treat each sample i as an anchor and every other pair (j, k) as candidates. Let $r_{ij} \in \{1, \ldots, B-1\}$ denote the rank of candidate j under the teacher's row $\tau_{i:}$ in descending order, and let $D(r) = 1/\log_2(1+r)$ denote the DCG discount (Järvelin & Kekäläinen, 2002). TIDE is trained with the weighted pairwise binary cross-entropy

$$\mathcal{L}_{\mathrm{rank}}(\rho, \tau) = \frac{1}{Z} \sum_i \sum_{\substack{j<k \\ j,k \neq i}} \left|D(r_{ij}) - D(r_{ik})\right| \cdot \mathrm{BCE}\Big(\mathrm{sigmoid}\big(\alpha \cdot (\rho_{ij} - \rho_{ik})\big), \mathbb{1}\left[\tau_{ij} > \tau_{ik}\right]\Big), \tag{9}$$

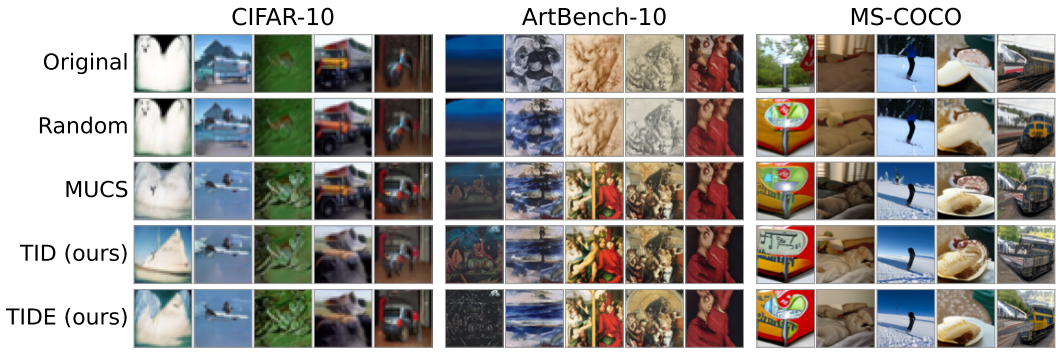


Figure 1: Qualitative comparison of five target queries from each dataset. Top row: original generations from the full-data model. Rows below: regenerations from the same initial noise after removing the samples attributed by each method. Random removal leaves regenerations nearly unchanged, whereas removal by MUCS, TID, and TIDE induces a visible drift.

where $\alpha > 0$ is a learnable global logit scale and $Z = \sum |D(r_{ij}) - D(r_{ik})|$ normalizes the weights. The DCG discount concentrates the loss on pairs near the top of the teacher's ranking, matching the top-k fidelity by which attribution is evaluated. Teacher quantities (τ and r) are detached, so only the embedding model receives gradients. Because Eq. 9 depends on embedding score differences, it is invariant to any per-anchor shift of ρ, leaving the logit scale α as the sole calibration parameter. Following the logit-scale treatment of CLIP (Radford et al., 2021), we parameterize it on a log scale.

Aligned multi-draw inference. Similar to when computing the expectation with TID (Eq. 5), we here also use multiple aligned draws at inference time and average ρ scores over M draws $\{(\sigma_m, \mathbf{n}_m)\}_{m=1}^{M}$ shared between the query and every training sample. Notice that the inference procedure can be efficient, as training-side embeddings can be computed once and cached. Each query requires only M forward passes at inference time (compared to the $|\mathcal{Z}| \cdot M$ passes required for the train set). Because the query embedding is constructed from the model's own activations, attribution is also available during generation itself (M is in the order of the number of diffusion steps and E_ϕ is much smaller than F_θ; we ablate M in Sec. 3.3 and provide architecture details in Appendix E).

3 Experiments and Results

3.1 Methodology

Following MUCS (Serrà et al., 2026), we evaluate attribution quality using *seed consistency*: diffusion models retrained after random data removal generate nearly identical images from the same initial noise, whereas removing samples that support a generation causes it to change. For each dataset and seed, we (i) train a model on the full dataset and generate a fixed set of queries; (ii) score all training samples for each query and remove the union of the per-query top-2%; (iii) retrain the model from scratch on the remaining data; and (iv) regenerate the queries using the same initial noise. Better attribution methods identify more influential samples and therefore induce greater deviation from the original generations.

We measure similarity between each original and regenerated image using SSIM (Wang et al., 2004), cosine similarity in the SSCD copy-detection space (Pizzi et al., 2022), LPIPS (Zhang et al., 2018), and cosine similarity in the CLIP semantic space (Radford et al., 2021). For each metric, we compare the similarities induced by a method's removals with those from an equally sized random-removal control, and summarize their separation using the area under the ROC curve (AUC). An AUC of 0.5 indicates performance equivalent to random removal, whereas 1 indicates perfect separation. We report the AUC for each metric, their mean μ, and the mean $\pm$ standard error over six different runs.

Data and models. We evaluate across diverse settings, including unconditional CIFAR-10 (Krizhevsky, 2009), class-conditional ArtBench-10 (Liao et al., 2022), and also text-conditioned MS-COCO (Lin et al., 2014) using CLIP embeddings. For each dataset, we train a DiT back-

Table 1: Counterfactual AUC per similarity metric and their average μ, reported as the mean over six seeds $\pm$ the standard error. Categories: MA = model-agnostic, G = gradient-based, U = unlearning-based, D = distillation-based. Best per column in **bold**, second best underlined.

	Approach	Categ.	SSIM	SSCD	LPIPS	CLIP	μ
CIFAR-10	CLIP	MA	0.598 $\pm$ 0.011	0.585 $\pm$ 0.008	0.610 $\pm$ 0.013	0.661 $\pm$ 0.010	0.613 $\pm$ 0.007
	DINO	MA	0.604 $\pm$ 0.018	0.616 $\pm$ 0.017	0.638 $\pm$ 0.021	0.665 $\pm$ 0.027	0.631 $\pm$ 0.019
	D-TRAK	G	0.691 $\pm$ 0.037	0.649 $\pm$ 0.033	0.703 $\pm$ 0.026	0.640 $\pm$ 0.042	0.671 $\pm$ 0.033
	DAS	G	0.760 $\pm$ 0.031	0.753 $\pm$ 0.036	0.789 $\pm$ 0.040	0.746 $\pm$ 0.035	0.762 $\pm$ 0.035
	MUCS	U	**0.864** $\pm$ 0.025	**0.854** $\pm$ 0.023	**0.908** $\pm$ 0.015	**0.857** $\pm$ 0.028	**0.870** $\pm$ 0.021
	TID (ours)	G	0.836 $\pm$ 0.031	0.815 $\pm$ 0.031	0.872 $\pm$ 0.023	0.818 $\pm$ 0.027	0.835 $\pm$ 0.027
	TIDE (ours)	D	0.856 $\pm$ 0.018	0.844 $\pm$ 0.017	0.901 $\pm$ 0.013	0.832 $\pm$ 0.013	0.858 $\pm$ 0.013
ArtBench-10	CLIP	MA	0.575 $\pm$ 0.036	0.575 $\pm$ 0.031	0.588 $\pm$ 0.036	0.621 $\pm$ 0.018	0.590 $\pm$ 0.029
	DINO	MA	0.558 $\pm$ 0.038	0.553 $\pm$ 0.028	0.573 $\pm$ 0.032	0.586 $\pm$ 0.034	0.567 $\pm$ 0.032
	D-TRAK	G	0.752 $\pm$ 0.035	0.707 $\pm$ 0.030	0.745 $\pm$ 0.031	0.677 $\pm$ 0.032	0.720 $\pm$ 0.029
	DAS	G	0.816 $\pm$ 0.020	0.794 $\pm$ 0.025	0.819 $\pm$ 0.021	0.787 $\pm$ 0.031	0.804 $\pm$ 0.022
	MUCS	U	0.867 $\pm$ 0.013	0.832 $\pm$ 0.018	0.869 $\pm$ 0.015	0.831 $\pm$ 0.012	0.850 $\pm$ 0.012
	TID (ours)	G	**0.905** $\pm$ 0.007	**0.870** $\pm$ 0.011	**0.918** $\pm$ 0.006	**0.861** $\pm$ 0.019	**0.889** $\pm$ 0.009
	TIDE (ours)	D	0.867 $\pm$ 0.009	0.835 $\pm$ 0.012	0.872 $\pm$ 0.018	0.825 $\pm$ 0.020	0.850 $\pm$ 0.011
MS-COCO	CLIP	MA	0.648 $\pm$ 0.022	0.678 $\pm$ 0.027	0.691 $\pm$ 0.028	0.799 $\pm$ 0.036	0.704 $\pm$ 0.026
	DINO	MA	0.683 $\pm$ 0.019	0.712 $\pm$ 0.032	0.718 $\pm$ 0.036	**0.811** $\pm$ 0.027	0.731 $\pm$ 0.027
	D-TRAK	G	0.696 $\pm$ 0.023	0.704 $\pm$ 0.033	0.746 $\pm$ 0.035	0.698 $\pm$ 0.031	0.711 $\pm$ 0.028
	DAS	G	0.718 $\pm$ 0.016	0.712 $\pm$ 0.033	0.761 $\pm$ 0.026	0.739 $\pm$ 0.035	0.733 $\pm$ 0.024
	MUCS	U	0.722 $\pm$ 0.019	0.717 $\pm$ 0.021	0.757 $\pm$ 0.025	0.755 $\pm$ 0.039	0.738 $\pm$ 0.022
	TID (ours)	G	**0.773** $\pm$ 0.019	**0.757** $\pm$ 0.022	**0.813** $\pm$ 0.029	0.788 $\pm$ 0.029	**0.783** $\pm$ 0.020
	TIDE (ours)	D	0.738 $\pm$ 0.025	0.727 $\pm$ 0.028	0.781 $\pm$ 0.025	0.748 $\pm$ 0.012	0.748 $\pm$ 0.018

bone (Peebles & Xie, 2023) under the EDM formulation (Karras et al., 2022). Full training, sampling, and evaluation details, along with an analysis of removal budgets, are provided in Appendix E.

Baselines. We compare against representative TDA methods from three broad families. *Model-agnostic* baselines retrieve samples by cosine similarity between CLIP (Radford et al., 2021) or DINO (Oquab et al., 2024) image embeddings. *Gradient-based* baselines include D-TRAK (Zheng et al., 2024) and DAS (Lin et al., 2025), which use (preconditioned) gradient similarity in a randomly projected parameter space. Although these methods were originally developed for DDPMs only, we adapt them to the EDM formulation while preserving their underlying design principles. As for *unlearning-based* baseline we consider MUCS (Serrà et al., 2026), which fine-tunes the model to unlearn each query and measures the resulting loss changes. We run all methods under the same evaluation protocol, using baseline hyperparameters adapted from their original settings (Appendix E).

3.2 Comparison with Existing Methods

Figure 1 illustrates the levels of degradation induced by removing samples identified by different methods. Under random removal, the regenerations are nearly indistinguishable from the originals, which illustrates the seed consistency that the evaluation relies on. Under TID and TIDE, the regenerations visibly drift in composition, style, or content, indicating that the removed samples indeed supported the queries. Additional qualitative results and discussion can be found in Appendix F.1.

Table 1 reports the counterfactual AUCs for the considered methods. TID attains the overall highest accuracy among the attribution methods: it surpasses MUCS on ArtBench-10 and MS-COCO while performing almost on par with it on CIFAR-10. The other existing gradient-based methods perform worse on all three datasets. TIDE, in turn, closely approaches the accuracy of its teacher TID, while requiring only forward passes and a lower dimensionality (we quantify this efficiency advantage in the following section). Additional quantitative results are available in Appendices F.2 and F.3.

3.3 Efficiency Trade-offs between TDA Methods

Attribution costs vary significantly (even at similar levels of performance, the methods in Table 1 differ by orders of magnitude in their per-query attribution cost). To characterize this efficacy–

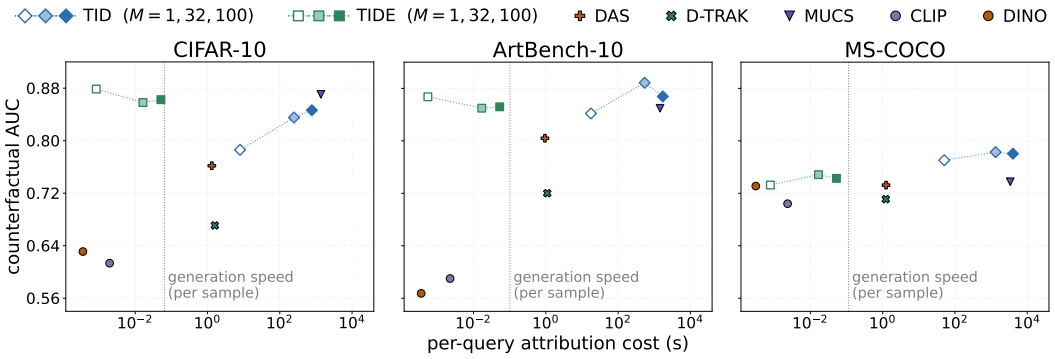


Figure 2: Attribution accuracy versus warm-cache per-query cost on CIFAR-10 (left), ArtBench (middle), and MS-COCO (right). The horizontal axis is logarithmic; cacheable training-side artifacts are precomputed. Markers with different fills connected by dotted lines trace the variation with M.

efficiency trade-off, we compare all methods on a common accuracy–cost plane. Figure 2 plots counterfactual AUCs against per-query cost. The methods essentially fall into two regimes. CLIP, DINO, DAS, and TIDE process only the query at attribution time and complete each attribution in microseconds to fractions of a second. In contrast, TID and MUCS repeat training-side computation for every query. TIDE alone occupies the upper-left region of the plane, combining the cost and storage profile of embedding-based methods with the accuracy of the expensive gradient- or unlearning-based ones. We note that TIDE can attribute a sample faster than the model generates it (vertical dotted lines in Figure 2). Appendix F.4 further provides an asymptotic cost analysis of each method, together with the one-off training-side and on-disk costs.

3.4 Dissecting the Design of TID

TID differs from existing gradient-based estimators along several dimensions (Sec. 2.1). We here isolate these design choices through five research questions. Specifically, we vary the scalar target being differentiated (RQ1), the corruption draws used to evaluate the query and training samples (RQ2), the normalization of per-sample gradients before kernel construction (RQ3), the terms retained from the exact counterfactual-retraining expansion (RQ4), and the curvature model used for preconditioning (RQ5). Table 2 reports the corresponding ablations on CIFAR-10.

RQ1: How much does the choice of attribution target matter? TID differs from existing gradient-based methods in its attribution target and curvature model. We first isolate the target by fixing the curvature model to that of TID and comparing its summed model output against the denoising loss $\omega(\sigma)\|\mathbf{x} - F_\theta\|_2^2$ (e.g., TRAK) and squared output norm $\|F_\theta\|_2^2$ (e.g., D-TRAK). We observe that both alternatives substantially reduce attribution accuracy, with the denoising loss causing the largest drop.

RQ2: Should corruption draws be aligned? Each gradient depends on the corruption draw at which it is evaluated. Although a similarity between gradients computed from independent draws is mathematically valid, it conflates differences between samples with differences between their corruptions. TID therefore samples one $(\sigma, \mathbf{n})$ per draw and shares it at inference and across the distillation batch. Replacing this shared draw with independent per-sample draws, while holding all other choices fixed, decreases μ by 9.2%. Independent draws introduce unmatched, noise-dependent components into the gradients, causing the kernel to capture variation in the corruption draws rather than the relationship between the samples.

RQ3: Should gradient magnitudes be retained? TID normalizes each sample's gradient factors before constructing the kernel, thereby discarding the gradient magnitude that a literal reading of Eq. 7 would retain. Restoring this magnitude reduces μ by 6.2%. At a given noise level, the gradient norm reflects how strongly the model responds to an individual sample, which depends on both sample difficulty and noise level. It is therefore a nuisance quantity for ranking training samples by their relationship to a query.

Table 2: Counterfactual AUCs for the ablation of the TID teacher on CIFAR-10. Each block isolates one design choice while holding the remaining components fixed. All variants use $M = 32$.

RQ/Variant	SSIM	SSCD	LPIPS	CLIP	μ	Δ
TID (ours)	0.836	0.815	0.872	0.818	0.835	—
RQ1: Attribution target						
$\|F_\theta\|_2^2$	0.815	0.797	0.855	0.790	0.814	-2.5%
$\omega(\sigma)\|\mathbf{x} - F_\theta\|_2^2$	0.777	0.737	0.785	0.731	0.757	-9.3%
RQ2: Corruption alignment						
Independent draws	0.759	0.743	0.798	0.732	0.758	-9.2%
RQ3: Gradient normalization						
+ Gradient magnitudes	0.795	0.769	0.822	0.747	0.783	-6.2%
RQ4: Newton corrections						
+ Residual r_i	0.835	0.822	0.868	0.805	0.832	-0.3%
+ Residual and leverage $r_i/(1-h_i)$	0.842	0.815	0.882	0.813	0.838	$+0.3\%$
RQ5: Curvature model						
+ MC-sampled GGN	0.792	0.773	0.838	0.767	0.792	-5.1%

RQ4: Do the exact one-step Newton corrections help? The one-step Newton expansion in Eq. 4 contains two sample-specific factors omitted by TID: the training residual r_i and the leverage $1 - h_i$, which discounts samples already well accommodated by the fit. We find that introducing either correction produces no meaningful improvement: the changes in μ are both below 1%. Thus, fidelity to the exact expansion does not appear to be a limiting factor, and the simpler kernel is preferable given its lower computational cost.

RQ5: Which curvature model should be used? TID preconditions its gradients using the Kronecker factors in Eq. 6, which are uncentered second moments of the observed activations and output gradients and thus constitute an empirical-Fisher approximation. In contrast, K-FAC Influence constructs a Monte Carlo-sampled generalized Gauss–Newton (GGN) matrix by resampling the target of a squared loss. Replacing the empirical Fisher with the sampled GGN decreases μ by 5.1%, which is smaller than the effect of changing the target in RQ1, indicating that methods' performance can be more affected by *what* they differentiate than *how* they approximate curvature.

4 Conclusion

We presented TID, a gradient-based TDA method for diffusion models that locally measures influence as the expected local discrepancy between the scores of the trained model and its leave-one-out counterfactual. Defined through the score, this target applies uniformly across diffusion variants, and a first-order expansion turns it into a Kronecker-factored gradient kernel that needs neither retraining, random projections, nor per-sample gradient storage. We further presented TIDE, which distills TID's rankings into embeddings computed from the diffusion model's own activations. With that, TDA reduces to cosine similarity on a low-dimensional space. Across three datasets, TID matches or surpasses unlearning-based attribution at a fraction of its cost, and TIDE retains most of this accuracy while attributing a query in milliseconds, faster than generating the sample itself. Our ablations also carry a broader lesson: the terms an exact derivation calls for, such as leverage corrections and schedule-dependent weights, consistently trade against ranking accuracy, and the simple aligned, normalized, uniformly aggregated kernel is the strongest practical choice.

Limitations and future work. We can enumerate three limitations of our work: (i) a general theory of when forward activations suffice to represent gradient-based influence is open; (ii) the distilled student is specific to the model and dataset it is trained on; (iii) our evaluation covers a single diffusion variant, although the formulation extends to DDPM, EDM, and flow matching. Besides limitations, we can also point to three directions for future work: (i) a distillation framework that transfers attribution knowledge across backbones, datasets, and teacher methods; (ii) stronger distribution-aware teachers, validated on the other diffusion variants; and (iii) reusing gradient information already computed during pre-training, so that teacher supervision and the student come at almost no additional cost.

AI USE STATEMENT

In this work, we have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, interpret results, assist with translation, clean and reformat datasets, or support qualitative and thematic data analysis. We have used generative AI tools to create or edit code, draft parts of the research paper, and edit the research paper to improve readability. We have reviewed all AI-assisted work: LLM-polished and LLM-drafted content was proofread and further improved by the authors, and LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human-subject studies, the collection of personal or sensitive data, or the creation or release of new datasets. It does not conduct interventions involving individuals or deploy systems to make decisions about them. Our methods aim to support transparency by estimating the influence of training examples on generated outputs. Practitioners should exercise caution when using these estimates to allocate compensation to data contributors or assess potential copyright infringement. Attribution scores alone do not establish copying or entitlement to compensation and should be considered alongside independent evidence and the relevant context.

REPRODUCIBILITY STATEMENT

All experiments use fixed random seeds, and results are reported over multiple seeds. Full experimental details are given in Appendix E, and the derivations supporting our method are given in Appendix C. We will release the code upon acceptance of the paper.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. RelatIF: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Jonathan Brokman, Omer Hofman, Roman Vainshtein, Amit Giloni, Toshiya Shimizu, Inderjeet Singh, Oren Rachmil, Alon Zolfi, Asaf Shabtai, Yuki Unno, et al. Montrage: Monitoring training for attribution of generative diffusion models. In *European Conference on Computer Vision (ECCV)*, 2024.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning (ICML)*, 2005.

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *USENIX Security Symposium (USENIX Security)*, 2023.

Zheng Dai and David K. Gifford. Training data attribution for diffusion models. *arXiv preprint arXiv:2306.02174*, 2023.

Junwei Deng, Yuzheng Hu, Pingbang Hu, Ting-Wei Li, Shixuan Liu, Jiachen T. Wang, Dan Ley, Qirun Dai, Benhao Huang, Jin Huang, et al. A survey of data attribution: Methods, applications, and evaluation in the era of generative AI. *SSRN preprint 5451054*, 2025.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a Kronecker-factored eigenbasis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Kristian Georgiev, Joshua Vendrow, Hadi Salman, Sung Min Park, and Aleksander Madry. The journey, not the destination: How data guides diffusion models. *arXiv preprint arXiv:2312.06205*, 2023.

Amirata Ghorbani and James Zou. Data Shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning (ICML)*, 2019.

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

Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. Evaluation of similarity-based explanations. In *International Conference on Learning Representations (ICLR)*, 2021.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. Data cleansing for models trained with SGD. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *International Conference on Machine Learning (ICML)*, 2022.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the Shapley value. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Myeongseob Ko, Feiyang Kang, Weiyan Shi, Ming Jin, Zhou Yu, and Ruoxi Jia. The mirrored influence hypothesis: Efficient data influence estimation by harnessing forward passes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

Zhifeng Kong and Kamalika Chaudhuri. Understanding instance-based interpretability of variational auto-encoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *arXiv preprint arXiv:2510.21890*, 2025.

Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The ArtBench dataset: Benchmarking generative models with artworks. *arXiv preprint arXiv:2206.11404*, 2022.

Jinxu Lin, Linwei Tao, Minjing Dong, and Chang Xu. Diffusion attribution score: Evaluating training data influence in diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision (ECCV)*, pp. 740–755, 2014.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Yiwei Lu, Matthew Y. R. Yang, Zuoqiu Liu, Gautam Kamath, and Yaoliang Yu. Disguised copyright infringement of latent diffusion models. In *International Conference on Machine Learning (ICML)*, 2024.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning (ICML)*, 2015.

Bruno Mlodozeniec, Runa Eschenhagen, Juhan Bae, Alexander Immer, David Krueger, and Richard E. Turner. Influence functions for scalable data attribution in diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

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.

W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2023.

Ed Pizzi, Sreya Dutta Roy, Sugosh Nagavara Ravindra, Priya Goyal, and Matthijs Douze. A self-supervised descriptor for image copy detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Daryl Pregibon. Logistic regression diagnostics. *The Annals of Statistics*, 9(4):705–724, 1981.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Christopher Scarvelis and Justin Solomon. Sensitivity analysis for diffusion models. *arXiv preprint arXiv: 2509.23092*, 2025.

Joan Serrà, Dipam Goswami, Fabio Morreale, Wei-Hsiang Liao, and Yuki Mitsufuji. Training data attribution in diffusion models via mirrored unlearning and noise-consistent skew. *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? Investigating data replication in diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Naoyuki Terashita, Hiroki Ohashi, Yuichi Nonaka, and Takashi Kanemaru. Influence estimation for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Jiachen T. Wang and Ruoxi Jia. Data Banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

Sheng-Yu Wang, Alexei A. Efros, Jun-Yan Zhu, and Richard Zhang. Evaluating data attribution for text-to-image models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Sheng-Yu Wang, Aaron Hertzmann, Alexei A. Efros, Jun-Yan Zhu, and Richard Zhang. Data attribution for text-to-image models by unlearning synthesized images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Sheng-Yu Wang, Aaron Hertzmann, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Fast data attribution for text-to-image models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The LambdaLoss framework for ranking metric optimization. In *ACM International Conference on Information and Knowledge Management (CIKM)*, 2018.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning (ICML)*, 2024.

Tong Xie, Haoyu Li, Andrew Bai, and Cho-Jui Hsieh. Data attribution for diffusion models: Timestep-induced bias in influence estimation. *Transactions on Machine Learning Research*, 2024.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Yang Zhang, Teoh Tze Tzun, Lim Wei Hern, and Kenji Kawaguchi. On copyright risks of text-to-image diffusion models. In *ECCV 2024 Workshop The Dark Side of Generative AIs and Beyond*, 2024.

Yutian Zhao, Chao Du, Xiaosen Zheng, Tianyu Pang, and Min Lin. Nonparametric data attribution for diffusion models. *International Conference on Machine Learning (ICML)*, 2026.

Xiaosen Zheng, Tianyu Pang, Chao Du, Jing Jiang, and Min Lin. Intriguing properties of data attribution on diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024.

A Related Work

Training data attribution. Training data attribution quantifies how training examples affect a target model behavior (Hammoudeh & Lowd, 2024; Deng et al., 2025). Major approaches include influence functions, which approximate leave-one-out effects using local parameter perturbations (Koh & Liang, 2017), weighted marginal-contribution methods such as Data Shapley and Data Banzhaf (Ghorbani & Zou, 2019; Jia et al., 2019; Wang & Jia, 2023), training-dynamics methods such as SGD-influence and TracIn (Hara et al., 2019; Pruthi et al., 2020), and simulators such as Datamodels, which learn model behavior from repeated subset retraining (Ilyas et al., 2022). TRAK combines ideas from influence functions and datamodels with random gradient projections to improve scalability in large, non-convex models (Park et al., 2023). These approaches expose a recurring trade-off: retraining-based methods provide direct counterfactual semantics but require many model fits, whereas gradient-based approximations reduce retraining at the cost of gradient computation, curvature estimation, and storage.

Attribution for diffusion models. Earlier work extends attribution to generative models including GANs and VAEs (Terashita et al., 2021; Kong & Chaudhuri, 2021). For diffusion models, ensemble-based methods enable explicit data ablation by training models on structured subsets (Dai & Gifford, 2023), while gradient-based methods adapt TRAK or influence functions to the denoising process. Among the latter, Journey-TRAK attributes intermediate denoising states (Georgiev et al., 2023), D-TRAK studies alternative diffusion-specific gradient targets (Zheng et al., 2024), Diffusion-ReTrac corrects timestep-induced bias (Xie et al., 2024), and DAS derives a diffusion-specific score with gradient normalization and KL-based motivation (Lin et al., 2025). Most closely related to TID, Mlodozeniec et al. (2025) use K-FAC curvature approximations to estimate influence on proxies for generation probability, including diffusion loss, ELBO, and trajectory probability. In contrast, TID instead defines influence directly as the expected score discrepancy at noised states of the query. A complementary line of work uses mirrored unlearning, estimating influence by unlearning the generated query and measuring the resulting changes on training samples (Ko et al., 2024; Wang et al., 2024; Serrà et al., 2026). These methods, however, require a separate optimization procedure for each query and cannot cache relevant information to reduce inference costs.

Efficient and representation-based attribution. Representation similarity supports efficient attribution through a single query encoding followed by nearest-neighbor retrieval, but generic visual similarity does not necessarily capture counterfactual influence (Georgiev et al., 2023; Wang et al., 2024). Prior work on representation-based attribution tunes visual representations using attribution-by-customization benchmarks (Wang et al., 2023), learns attribution models from representation changes during diffusion fine-tuning (Brokman et al., 2024), or develops nonparametric attribution from representation geometry (Zhao et al., 2026). Most closely related to TIDE, Fast Data Attribution distills an unlearning-based teacher into a standalone visual encoder using an offline collection of annotated pairs (Wang et al., 2025). In contrast to this, TIDE generates dense TID supervision online within each training batch and embeds the diffusion model's own internal activations. It directly distills per-query rankings rather than raw influence values, producing compact embeddings that retain model-specific attribution information while enabling efficient nearest-neighbor retrieval.

A.1 Details of Gradient-based Methods

Most gradient-based attribution methods for diffusion models, including ours, follow a common template: differentiate a scalar function of the model evaluated on a corrupted training sample, incorporate a curvature approximation, and compare each training sample with the query through a curvature-aware gradient inner product. As mentioned in Sec. 2.1, gradient-based methods differ primarily in the scalar target being differentiated, the curvature approximation, and the compression and normalization applied to the per-sample gradients. Table 3 organizes existing methods and the ablations in Table 2 along these design axes.

B An Overview of TID and TIDE

Figure 3 illustrates how the teacher of Sec. 2.1 and the student of Sec. 2.2 share a single pass through the frozen diffusion model. Panel (a) shows one distillation step of Algorithm 1. A minibatch is

Table 3: Design choices of gradient-based diffusion attribution methods: D-TRAK (Zheng et al., 2024), DAS (Lin et al., 2025), K-FAC influence (Mlodozeniec et al., 2025), and TID. Here, F_θ denotes the denoiser output evaluated on a corrupted sample and $\mathbf{x}$ denotes the corresponding clean sample. Curvature specifies the preconditioner; kernel specifies how per-sample gradients are represented before computing inner products; and normalization specifies which representation, if any, is normalized at each timestep. Gradient normalization rescales each projected gradient to unit norm, whereas factor normalization rescales the two Kronecker factors independently, equivalently normalizing the corresponding rank-one gradient matrix by its Frobenius norm. Without normalization, large-gradient samples can dominate the aggregation across timesteps. The lower block varies individual components of TID and is evaluated quantitatively in Table 2.

Method	Target	Curvature	Kernel	Normalization
D-TRAK	$\|F_\theta\|_2^2$	GN + ridge	random projection	none
DAS	$\omega(\sigma)\|\mathbf{x} - F_\theta\|_2^2$	GN + ridge	random projection	gradient
K-FAC influence	$\omega(\sigma)\|\mathbf{x} - F_\theta\|_2^2$	MC-GGN	K-FAC	none
TID (ours)	$\langle \mathbf{1}, F_\theta \rangle$	empirical Fisher	K-FAC	factors
Ablation variants of TID				
RQ1a: squared output	$\|F_\theta\|_2^2$	empirical Fisher	K-FAC	factors
RQ1b: denoising loss	$\omega(\sigma)\|\mathbf{x} - F_\theta\|_2^2$	empirical Fisher	K-FAC	factors
RQ2: independent draws	$\langle \mathbf{1}, F_\theta \rangle$	empirical Fisher	K-FAC	factors
RQ3: gradient magnitude	$\langle \mathbf{1}, F_\theta \rangle$	empirical Fisher	K-FAC	**none**
RQ4a: $+ r_i$	$\langle \mathbf{1}, F_\theta \rangle$	empirical Fisher	K-FAC	factors
RQ4b: $+ r_i/(1-h_i)^2$	$\langle \mathbf{1}, F_\theta \rangle$	empirical Fisher	K-FAC	factors
RQ5: sampled curvature	$\langle \mathbf{1}, F_\theta \rangle$	**MC-GGN**	K-FAC	factors

noised with one shared draw $(\sigma, \mathbf{n})$ under the aligned regime, and passed through F_θ once. Along the gradient path, the backward pass yields the per-layer factors $(\mathbf{a}_{l,i}, \mathbf{b}_{l,i})$, which the Kronecker-factored kernel of Eq. 6 turns into the within-batch teacher matrix $\boldsymbol{\tau}$ of TID scores. Along the activation path, the residual streams $\mathbf{h}_1, \ldots, \mathbf{h}_K$ of the same forward pass are summed, attention-pooled, and projected into the TIDE embeddings of Eq. 8, whose cosine similarities form the student matrix $\boldsymbol{\rho}$. The student and teacher matrices are compared row by row by the ranking loss of Eq. 9, with each sample acting as anchor and the diagonal excluded. The teacher is detached, so only the embedding model E_ϕ is updated. Panel (b) shows attribution at inference time with TIDE: a generated query is noised with the M aligned draws, embedded from forward activations only, and scored by cosine similarity against training embeddings that are computed once and cached, so no backward pass is needed.

Distillation Algorithm. Algorithm 1 gives the procedural form of the distillation step illustrated in Figure 3(a). The Kronecker factor inverses of Eq. 6 are estimated once from the training set before distillation begins and are held constant, so the only quantities recomputed per batch are the per-sample gradient factors and embeddings. Each batch uses a single shared draw $(\sigma, \mathbf{n})$ under the aligned regime of Sec. 2.1, and one forward–backward pass per sample supplies the information for both paths: the residual-stream activations feed the embedding of Eq. 8 and the per-layer gradient factors feed the teacher kernel of Eq. 6. The teacher matrix $\boldsymbol{\tau}$ is detached, so the gradient step on the ranking loss of Eq. 9 updates only the embedding parameters ϕ while the diffusion model F_θ is never modified. The per-batch cost is therefore that of one backward pass per sample plus $O(B^2)$ inner products, and no teacher scores are stored across batches.

C Handling Multiple Diffusion Variants

C.1 Theoretical Background

We briefly recall the SDE view of diffusion models (Song et al., 2021b), indexed directly by the noise scale σ used throughout the paper. We first fix the notation of the expectations used below, which we keep out of the main text. The expectation $\mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}$ is over a training instances $\mathbf{z} = \{\mathbf{x}, \mathbf{c}\}$ drawn from $\mathcal{Z}$, noise scales σ drawn from the variant's training noise-scale distribution, and noise vectors $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, I)$. In $\mathbb{E}_{\sigma,\mathbf{n}}$ we omit the draw of $\mathbf{z}$. The network $F_\theta(\mathbf{x}_{\sigma,\mathbf{n}}; \sigma, \mathbf{c})$ is evaluated on the

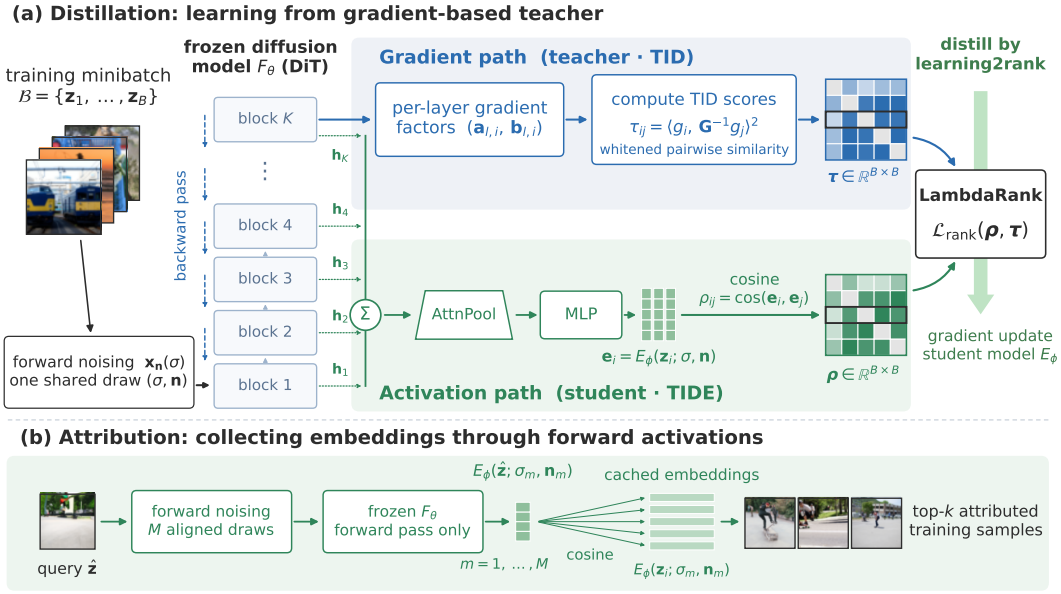


Figure 3: Overview of TID and TIDE.

Algorithm 1 Batched pair-wise score distillation

Require: frozen diffusion model F_θ; training set $\mathcal{Z}$; student embedder E_ϕ; K-FAC factor inverses $\{\mathbf{A}_l^{-1}, \mathbf{B}_l^{-1}\}$ estimated once and offline from $\mathcal{Z}$ (Sec. 2.1).

for each training batch $\mathcal{B} = \{\mathbf{z}_1, \ldots, \mathbf{z}_B\} \subset \mathcal{Z}$ **do**
  draw one shared $(\sigma, \mathbf{n})$ with σ from the sampler's schedule and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, I)$
  for each $\mathbf{z}_i = \{\mathbf{x}, \mathbf{c}\} \in \mathcal{B}$ **do**
    forward–backward pass of $F_\theta(\mathbf{x}_{\sigma,\mathbf{n}}; \sigma, \mathbf{c})$
    collect the activations $\{\mathbf{h}_k(\mathbf{z}_i; \sigma, \mathbf{n})\}_{k=1}^K$ and compute embeddings $\mathbf{e}_i$ using Eq. 8
    collect the gradient factors $(\mathbf{a}_{l,i}, \mathbf{b}_{l,i})$
  end for
  for all pairs (i, j) in $\mathcal{B}$ **do**
    compute teacher scores τ_{ij} (detached) using gradient factors $\{(\mathbf{a}_{l,i}, \mathbf{b}_{l,i}), (\mathbf{a}_{l,j}, \mathbf{b}_{l,j})\}$ and Eq. 6
    compute student scores ρ_{ij} using embeddings $(\mathbf{e}_i, \mathbf{e}_j)$ and cosine similarity
  end for
  update ϕ by one gradient step on $\mathcal{L}_{\text{rank}}(\rho, \tau)$ using Eq. 9
end for
return E_ϕ

forward-noised item $\mathbf{x}_{\sigma,\mathbf{n}}$, defined below, with the noise scale σ and the conditioning $\mathbf{c}$ as additional inputs. All densities in this section are conditioned on $\mathbf{c}$, which we suppress when unambiguous.

Forward and reverse processes. Given a training instance $\mathbf{z} = \{\mathbf{x}, \mathbf{c}\}$, a diffusion model specifies a stochastic process $\{\mathbf{x}_\sigma\}_{\sigma \in [0, \sigma_{\max}]}$ that starts at the data item, $\mathbf{x}_\sigma = \mathbf{x}$ at $\sigma = 0$, and ends at a tractable prior at $\sigma_{\max}$, by solving the Itô SDE

$$d\mathbf{x}_\sigma = \mathbf{f}(\mathbf{x}_\sigma, \sigma)\, d\sigma + \eta(\sigma)\, d\mathbf{W}_\sigma,$$

where $\mathbf{f}(\cdot, \sigma) : \mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient, $\eta : \mathbb{R} \to \mathbb{R}$ the scalar diffusion coefficient, and $\mathbf{W}_\sigma$ a standard Wiener process. We write $q(\mathbf{x}_\sigma \mid \mathbf{x})$ for the transition kernel of this process from the clean item $\mathbf{x}$ to noise scale σ, and p_σ for the resulting marginal of $\mathbf{x}_\sigma$. Anderson (1982) shows that the reversal of this process is itself a diffusion

$$d\mathbf{x}_\sigma = \left[\mathbf{f}(\mathbf{x}_\sigma, \sigma) - \eta(\sigma)^2\, \nabla_{\mathbf{x}_\sigma} \log p_\sigma(\mathbf{x}_\sigma)\right] d\sigma + \eta(\sigma)\, d\bar{\mathbf{W}}_\sigma,$$

where $\bar{\mathbf{W}}_\sigma$ is a standard Wiener process when σ runs backwards from $\sigma_{\max}$ to 0 and $d\sigma$ is an infinitesimal negative step. Generation thus reduces to knowing the score $\nabla_{\mathbf{x}_\sigma} \log p_\sigma(\mathbf{x}_\sigma)$ at every

noise scale. The samplers we consider discretize this reverse process, or its deterministic probability-flow counterpart (Song et al., 2021b), on the schedule $\sigma_{\max} = \sigma_T > \cdots > \sigma_1 = \sigma_{\min}$ of Sec. 2.

Score matching. The score $\nabla_{\mathbf{x}_\sigma} \log p_\sigma(\mathbf{x}_\sigma)$ is learned by a noise-conditioned network s_θ via denoising score matching,

$$\min_\theta \; \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}\Big[\lambda(\sigma)\,\big\|s_\theta(\mathbf{x}_{\sigma,\mathbf{n}};\sigma,\mathbf{c}) - \nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma \mid \mathbf{x})\big|_{\mathbf{x}_\sigma=\mathbf{x}_{\sigma,\mathbf{n}}}\big\|_2^2\Big],$$

where $\lambda : [\sigma_{\min}, \sigma_{\max}] \to \mathbb{R}_{>0}$ is a weighting function and $\mathbf{x}_{\sigma,\mathbf{n}} \sim q(\cdot \mid \mathbf{x})$. For any positive λ, the minimizer equals the marginal score almost everywhere. The objective only requires the transition kernel q. When the drift is affine and has the scalar linear form $\mathbf{f}(\mathbf{x}_\sigma, \sigma) = f(\sigma)\mathbf{x}_\sigma$, as in every variant we consider, the transition kernel is Gaussian,

$$q(\mathbf{x}_\sigma \mid \mathbf{x}) = \mathcal{N}\big(\mu(\sigma)\mathbf{x}, \nu(\sigma)^2 I\big),$$

with closed-form signal and noise coefficients

$$\mu(\sigma) = \exp\left(\int_0^\sigma f(u)\,du\right) \qquad \text{and} \qquad \nu(\sigma)^2 = \mu(\sigma)^2 \int_0^\sigma \frac{\eta(u)^2}{\mu(u)^2}\,du,$$

so that a forward-noised state and the corresponding score-matching target can be written directly in terms of the noise draw $\mathbf{n}$. Using the notation of Sec. 2 the forward-noised state is

$$\mathbf{x}_{\sigma,\mathbf{n}} = \mu(\sigma)\,\mathbf{x} + \nu(\sigma)\,\mathbf{n},$$

and the score-matching target is

$$\nabla_{\mathbf{x}_\sigma} \log q(\mathbf{x}_\sigma \mid \mathbf{x})\Big|_{\mathbf{x}_\sigma=\mathbf{x}_{\sigma,\mathbf{n}}} = -\frac{\mathbf{x}_{\sigma,\mathbf{n}} - \mu(\sigma)\,\mathbf{x}}{\nu(\sigma)^2} = -\frac{\mathbf{n}}{\nu(\sigma)}.$$

For the variance-exploding SDE, $\mathbf{f}(\mathbf{x}_\sigma, \sigma) = \mathbf{0}$ and $\eta(\sigma) = \sqrt{2\sigma}$ gives $(\mu, \nu) = (1, \sigma)$, and this becomes the process underlying EDM (Karras et al., 2022). For the variance-preserving SDE, $\mathbf{f}(\mathbf{x}_\sigma, \sigma) = -\frac{1}{2}\beta(\sigma)\,\mathbf{x}_\sigma$ and $\eta(\sigma) = \sqrt{\beta(\sigma)}$ for a positive noise schedule β gives $(\mu, \nu) = (\sqrt{\alpha(\sigma)}, \sqrt{1-\alpha(\sigma)})$ with $\alpha(\sigma) = \exp\big(-\int_0^\sigma \beta(s)\,ds\big)$, and this becomes the continuous limit of DDPM (Ho et al., 2020). The linear interpolant of flow matching, $(\mu, \nu) = (1-\sigma, \sigma)$ on $\sigma \in (0, 1)$, is usually presented as an ODE rather than an SDE, but has the same Gaussian conditional kernel, so its conditional score is given by the same expression.

From network output to score. In practice, the network F_θ is not trained to output the score itself but a schedule-dependent regression target, such as the clean instance, the noise, or a velocity. The network is evaluated on the forward-noised instance as $F_\theta(\mathbf{x}_{\sigma,\mathbf{n}}; \sigma, \mathbf{c})$, with the noise scale σ and the conditioning $\mathbf{c}$ as additional inputs, and $\hat{\mathbf{x}}_{\sigma,\mathbf{n}}$ denotes the forward noising of an item $\mathbf{x}$ with the same draw $(\sigma, \mathbf{n})$. In this notation, the training loss L implements the diffusion variant together with its forward noising.

For DDPM (Ho et al., 2020), F_θ predicts the noise,

$$L(F_\theta) = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}\Big[\big\|\mathbf{n} - F_\theta\big(\mathbf{x}_{\sigma,\mathbf{n}};\sigma,\mathbf{c}\big)\big\|_2^2\Big] \quad \text{with} \quad \mathbf{x}_{\sigma,\mathbf{n}} = \sqrt{\alpha(\sigma)}\,\mathbf{x} + \sqrt{1-\alpha(\sigma)}\,\mathbf{n}.$$

For EDM (Karras et al., 2022), F_θ predicts the clean instance under a noise-scale-dependent weight ω,

$$L(F_\theta) = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}\Big[\omega(\sigma)\,\big\|\mathbf{x} - F_\theta\big(\mathbf{x}_{\sigma,\mathbf{n}};\sigma,\mathbf{c}\big)\big\|_2^2\Big] \quad \text{with} \quad \mathbf{x}_{\sigma,\mathbf{n}} = \mathbf{x} + \sigma\mathbf{n}.$$

For flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023), F_θ regresses the conditional velocity $\mathbf{n} - \mathbf{x}$ along the linear path, with $\sigma \sim \mathcal{U}(0, 1)$,

$$L(F_\theta) = \mathbb{E}_{\mathbf{z},\sigma,\mathbf{n}}\Big[\big\|\mathbf{n} - \mathbf{x} - F_\theta\big(\mathbf{x}_{\sigma,\mathbf{n}};\sigma,\mathbf{c}\big)\big\|_2^2\Big] \quad \text{with} \quad \mathbf{x}_{\sigma,\mathbf{n}} = (1-\sigma)\,\mathbf{x} + \sigma\,\mathbf{n}.$$

Each target determines the noise $\mathbf{n}$ as a function of F_θ and the noisy state $\mathbf{x}_{\sigma,\mathbf{n}}$, so the score estimate s_θ is recovered by substitution into the conditional score $-\mathbf{n}/\nu(\sigma)$ derived above, and for all common parameterizations the resulting map is affine.

Lemma 1 (Affine output-to-score map; Ho et al., 2020; Karras et al., 2022)**.** *For the common diffusion parameterizations, the score estimate is an affine function of the network output with coefficients depending only on the noise schedule,*

$$s_\theta = \kappa(\sigma) F_\theta + b(\sigma)\mathbf{x}.$$

For the standard DDPM ϵ-prediction parameterization,

$$s_\theta = -\frac{F_\theta}{\sqrt{1-\alpha(\sigma)}}, \qquad \kappa(\sigma) = -\frac{1}{\sqrt{1-\alpha(\sigma)}}, \qquad b(\sigma) = 0,$$

while for EDM,

$$s_\theta = \frac{F_\theta - \mathbf{x}}{\sigma^2}, \qquad \kappa(\sigma) = \frac{1}{\sigma^2}, \qquad b(\sigma) = -\frac{1}{\sigma^2},$$

and for flow matching on the linear path $\mathbf{x}_{\sigma,\mathbf{n}} = (1-\sigma)\mathbf{x} + \sigma\mathbf{n}$, where F_θ predicts the velocity $\mathbf{n} - \mathbf{x}$ so that $\mathbf{n} = \mathbf{x}_{\sigma,\mathbf{n}} + (1-\sigma)F_\theta$,

$$s_\theta = -\frac{(1-\sigma)\, F_\theta + \mathbf{x}_{\sigma,\mathbf{n}}}{\sigma}, \qquad \kappa(\sigma) = -\frac{1-\sigma}{\sigma}, \qquad b(\sigma) = -\frac{1}{\sigma}.$$

C.2 Variant Weights

This section instantiates the conversion weight $w_\sigma = \kappa(\sigma)^2$ of Eq. 1 for the common parameterizations. By Lemma 1, the score estimate is affine in the network output, $s_\theta = \kappa(\sigma)F_\theta + b(\sigma)\mathbf{x}$. Since the term $b(\sigma)\mathbf{x}$ is shared by any two models evaluated at the same state, it cancels when computing score differences, giving

$$\|s_\theta - s_{\theta^\setminus}\|_2^2 = \kappa(\sigma)^2 \, \|F_\theta - F_{\theta^\setminus}\|_2^2$$

and hence we obtain Eq. 1 with $w_\sigma = \kappa(\sigma)^2$. The coefficients follow from expressing the conditional score $\nabla_{\mathbf{x}_\mathbf{n}} \log q(\mathbf{x}_\mathbf{n} \,|\, \mathbf{x})$ in each variant's output parameterization.

- EDM: F_θ predicts the clean item and $s_\theta = (F_\theta - \mathbf{x}_\mathbf{n})/\sigma^2$, so

$$\kappa(\sigma) = \frac{1}{\sigma^2} \qquad \text{and} \qquad w_\sigma = \frac{1}{\sigma^4}.$$

- DDPM (ϵ-prediction): $s_\theta = -F_\theta/\sqrt{1-\alpha(\sigma)}$, so

$$\kappa(\sigma) = -\frac{1}{\sqrt{1-\alpha(\sigma)}} \qquad \text{and} \qquad w_\sigma = \frac{1}{1-\alpha(\sigma)}.$$

- $\mathbf{v}$-prediction: With $\mathbf{x}_\mathbf{n} = \sqrt{\alpha}\,\mathbf{x} + \sqrt{1-\alpha}\,\mathbf{n}$ and F_θ predicting $\mathbf{v} = \sqrt{\alpha}\,\mathbf{n} - \sqrt{1-\alpha}\,\mathbf{x}$, the identities $\mathbf{n} = \sqrt{1-\alpha}\,\mathbf{x}_\mathbf{n} + \sqrt{\alpha}\,F_\theta$ and $s_\theta = -\mathbf{n}/\sqrt{1-\alpha}$ give

$$\kappa(\sigma) = -\frac{\sqrt{\alpha(\sigma)}}{\sqrt{1-\alpha(\sigma)}} \qquad \text{and} \qquad w_\sigma = \frac{\alpha(\sigma)}{1-\alpha(\sigma)},$$

  the signal-to-noise ratio.

- Flow matching: For the linear path $\mathbf{x}_\mathbf{n}(\sigma) = (1-\sigma)\mathbf{x} + \sigma\mathbf{n}$, $\sigma \in (0,1)$, with velocity output F_θ regressing $\mathbf{n} - \mathbf{x}$, eliminating $\mathbf{x}$ via $\mathbf{x} = \mathbf{x}_\mathbf{n} - \sigma F_\theta$ gives $\mathbf{n} = \mathbf{x}_\mathbf{n} + (1-\sigma)F_\theta$; the conditional score of the path's Gaussian kernel then yields

$$\kappa(\sigma) = -\frac{1-\sigma}{\sigma} \qquad \text{and} \qquad w_\sigma = \frac{(1-\sigma)^2}{\sigma^2},$$

  up to the path's noise normalization.

In all cases, w_σ depends only on the noise schedule and is bounded above and below by positive constants on the truncated window $[\sigma_{\min}, \sigma_{\max}]$ with $\sigma_{\min} > 0$ (and $\sigma_{\max} < 1$ for flow matching).

D FACTORIZED GRADIENTS BEYOND LINEAR LAYERS

The main text presents the K-FAC decomposition for a linear layer without a token dimension. Here, we give the corresponding formulas for the other parameterized layers commonly used in diffusion architectures. Throughout this section, $\mathbf{a}_{l,i,t} \in \mathbb{R}^{N_{l,\text{in}}}$ and $\mathbf{b}_{l,i,t} \in \mathbb{R}^{N_{l,\text{out}}}$ denote the two gradient factors for training sample $\mathbf{z}_i$ at position t. The position index may represent a sequence token or a spatial location. The per-example layer gradient has the general form

$$g_l(\mathbf{z}_i) = \sum_{t=1}^{T_l} \mathbf{b}_{l,i,t}\mathbf{a}_{l,i,t}^\top. \tag{10}$$

Consequently, the inner product between two layer gradients can be evaluated without materializing either gradient:

$$\big\langle g_l(\mathbf{z}_i), g_l(\mathbf{z}_j)\big\rangle = \sum_{t=1}^{T_l}\sum_{s=1}^{T_l} \big(\mathbf{a}_{l,i,t}^\top \mathbf{a}_{l,j,s}\big)\big(\mathbf{b}_{l,i,t}^\top \mathbf{b}_{l,j,s}\big). \tag{11}$$

The corresponding squared norm is obtained by setting $i = j$. Under K-FAC, positions are treated as observations, giving

$$\mathbf{A}_l = \frac{1}{BT_l}\sum_{i,t} \mathbf{a}_{l,i,t}\mathbf{a}_{l,i,t}^\top, \qquad \mathbf{B}_l = \frac{1}{BT_l}\sum_{i,t} \mathbf{b}_{l,i,t}\mathbf{b}_{l,i,t}^\top. \tag{12}$$

The preconditioned inner product therefore remains factorized:

$$\big\langle g_l(\mathbf{z}_i), \mathbf{G}_l^{-1} g_l(\mathbf{z}_j)\big\rangle \approx \sum_{t=1}^{T_l}\sum_{s=1}^{T_l} \big(\mathbf{a}_{l,i,t}^\top \mathbf{A}_l^{-1}\mathbf{a}_{l,j,s}\big)\big(\mathbf{b}_{l,i,t}^\top \mathbf{B}_l^{-1}\mathbf{b}_{l,j,s}\big). \tag{13}$$

Linear layers with token dimensions. For a linear transformation applied independently to T_l tokens, $\mathbf{a}_{l,i,t}$ is the input activation and $\mathbf{b}_{l,i,t}$ is the gradient with respect to the pre-activation output. Its weight gradient is exactly Eq. 10. When no token dimension is present, $T_l = 1$, and Eq. 13 reduces to Eq. 6 in the main text. A bias can be included by augmenting every activation factor as $\tilde{\mathbf{a}}_{l,i,t} = [\mathbf{a}_{l,i,t}^\top, 1]^\top$.

Convolutional layers. Consider a two-dimensional convolution with kernel size $K_h \times K_w$ and C_{in} input channels. After applying the standard *unfold* (im2col) operator, let

$$\mathbf{a}_{l,i,t} \in \mathbb{R}^{C_{\text{in}}K_hK_w}, \qquad \mathbf{b}_{l,i,t} \in \mathbb{R}^{C_{\text{out}}}, \qquad T_l = H_{\text{out}}W_{\text{out}}, \tag{14}$$

where $\mathbf{a}_{l,i,t}$ is the input patch associated with output location t and $\mathbf{b}_{l,i,t}$ is the corresponding pre-activation output gradient. The flattened kernel gradient is

$$g_l(\mathbf{z}_i) = \sum_{t=1}^{T_l} \mathbf{b}_{l,i,t}\mathbf{a}_{l,i,t}^\top, \tag{15}$$

so Eqs. 11–13 apply directly. The same construction extends to one- and three-dimensional convolutions by replacing the receptive-field dimension accordingly. Convolutional biases are again incorporated by appending a constant to each unfolded input patch.

Transposed convolutional layers. For a transposed convolution, the roles of the unfolded quantities are reversed. Let t index the T_l input spatial locations, let $\mathbf{a}_{l,i,t} \in \mathbb{R}^{C_{\text{in}}}$ be the input at location t, and let $\mathbf{b}_{l,i,t} \in \mathbb{R}^{C_{\text{out}}K_hK_w}$ be the corresponding kernel-shaped patch extracted from the output gradient. Up to the fixed permutation induced by the kernel layout,

$$g_l(\mathbf{z}_i) = \sum_{t=1}^{T_l} \mathbf{a}_{l,i,t}\mathbf{b}_{l,i,t}^\top. \tag{16}$$

Because transposition does not change a Frobenius inner product, the same ghost-inner-product and K-FAC formulas apply after exchanging the two factors. Unlike an ordinary convolution, the bias

dimension C_{out} does not match the unfolded output-gradient dimension $C_{\text{out}}K_hK_w$; we therefore handle the bias as a separate parameter block, with per-example gradient

$$g_{l,\text{bias}}(\mathbf{z}_i) = \sum_u \frac{\partial \mathcal{L}_i}{\partial \mathbf{y}_{l,i,u}}. \tag{17}$$

This avoids introducing padded factors solely to combine the weight and bias blocks.

Normalization layers. For LayerNorm with scale $\boldsymbol{\gamma}_l$ and shift $\boldsymbol{\beta}_l$,

$$\mathbf{y}_{l,i,t} = \boldsymbol{\gamma}_l \odot \hat{\mathbf{x}}_{l,i,t} + \boldsymbol{\beta}_l, \tag{18}$$

where $\hat{\mathbf{x}}_{l,i,t}$ is the normalized input. Writing $\mathbf{b}_{l,i,t} = \partial \mathcal{L}_i / \partial \mathbf{y}_{l,i,t}$, the per-example parameter gradients are

$$g_{l,\gamma}(\mathbf{z}_i) = \sum_t \hat{\mathbf{x}}_{l,i,t} \odot \mathbf{b}_{l,i,t}, \qquad g_{l,\beta}(\mathbf{z}_i) = \sum_t \mathbf{b}_{l,i,t}. \tag{19}$$

Thus, their exact inner product is

$$\big\langle g_l(\mathbf{z}_i), g_l(\mathbf{z}_j) \big\rangle = g_{l,\gamma}(\mathbf{z}_i)^\top g_{l,\gamma}(\mathbf{z}_j) + g_{l,\beta}(\mathbf{z}_i)^\top g_{l,\beta}(\mathbf{z}_j). \tag{20}$$

These gradients are vectors rather than outer products, so we use a diagonal curvature approximation for normalization parameters. The same elementwise formulation applies to other affine normalization layers, including GroupNorm and InstanceNorm, after summing over the axes on which their scale and shift parameters are shared.

Together, these constructions cover the principal parameterized layer types in the diffusion architectures used in our experiments while preserving factorized computation wherever an outer-product structure is available.

E Experimental Details

Models, training, and compute. All experiments are run on a single NVIDIA H100 80 GB GPU. Table 4 summarizes the datasets, model architectures, training configurations, and evaluation protocol, while Table 5 reports the settings of TID and TIDE. All diffusion models use DiT backbones trained under the EDM formulation, following the code released with MUCS (Serrà et al., 2026). We use an exponential moving average of the diffusion model parameters during training. Each counterfactual model is retrained with the same configuration and random seed as its full-data counterpart, excluding only the selected influential samples. This paired design reduces variation due to optimization and isolates the effect of removing the attributed data.

Protocol details. For each seed, we generate a fixed set of 20 held-out queries using the full-data model. Every TDA method selects the top 2% of the training set for each query, after which we retrain the model without the union of the 20 selected sets. Because the per-query sets overlap, the number of distinct samples removed depends on the method. Rankings that repeatedly select the same samples yield a smaller union than rankings that are more query-specific. As shown in Table 6, the resulting removal fractions range from 21% to 33%. The random control produces the largest union because its per-query selections overlap the least, whereas the model-agnostic encoders tend to retrieve visually central samples shared across queries and therefore produce smaller unions. Appendix F.2 examines sensitivity to the per-query budget. We regenerate each query from the same initial latent used by the full-data model. For each metric, we compute the AUC of each run's query–regeneration similarities against its equally sized random-removal control, and report the mean and standard error of the resulting AUCs over the six runs.

Method settings. Table 5 lists the complete method configurations. TID uses $M = 32$ timesteps from the Karras schedule on every dataset, matching the number of sampling steps. Figure 2 additionally evaluates $M \in \{1, 32, 100\}$. We prenormalize each sample's factors and invert the K-FAC factors with relative damping $\alpha = 0.1$, setting the damping parameter for each factor to 0.1 times its mean eigenvalue. For the architecture of TIDE (Eq. 8), $\mathrm{AttnPool}$ is a four-head attention layer with a single learned query token, which pools a token sequence into one 768-dimensional vector, and MLP is a two-layer perceptron ($768 \to 512 \to 768$, SiLU activation). Together they hold 3.2M trainable parameters (2.4M and 0.8M, respectively), which is only around 3% of the $\sim$100M-parameter backbone. The same architecture is used unchanged on all three datasets. We train TIDE

Table 4: Dataset, model, training, sampling, and evaluation configurations. Attribution-specific settings are reported separately in Table 5.

		CIFAR-10	ArtBench-10	MS-COCO
Data	Training samples N	50,000	49,917	118,287
	Resolution	32×32	64×64	64×64
	Conditioning	none	class label	CLIP text embedding
Backbone	Architecture	DiT	DiT	DiT
	Transformer blocks	12	12	12
	Model width	768	768	768
	Attention heads	12	12	12
	Patch size	2×2	2×2	2×2
	Stem channels	$\{4, 4\}$	$\{16, 16\}$	$\{16, 16\}$
	Stem resampling	$\{1, 1\}$	$\{1, 2\}$	$\{1, 2\}$
	Conditioning width	256	256	1,024
Optimization	Optimizer	AdamW	AdamW	AdamW
	Learning rate	1×10^{-4}	2×10^{-4}	2×10^{-4}
	Weight decay	0.01	0.01	0.01
	Batch size	128	128	128
	Updates per epoch	1,000	1,000	1,000
	Epochs	200	250	300
	EMA decay	0.999	0.999	0.999
Sampler	Schedule	Karras	Karras	Karras
	Steps	32	32	32
	σ range	$[0.002, 80]$	$[0.002, 80]$	$[0.002, 80]$
	ρ	7	7	7
Protocol	Queries per seed	20	20	20
	Seeds	6	6	6
	Per-query budget	2%	2%	2%

for 50 epochs using LambdaRank distillation and a cosine-similarity head. The similarity scale is parameterized as $\alpha = e^{\beta}$, with β initialized to $\log 32$, and optimized using the base learning rate of 5×10^{-5}. At inference, TIDE also averages scores over M aligned draws. Table 1 uses $M = 32$, while Fig. 2 evaluates $M \in \{1, 32, 100\}$; variation across these settings is smaller than the variation across seeds. Baseline hyperparameters follow their official implementations. For DAS, we average projected gradients across timesteps after normalizing them independently at each timestep, use a projection dimension of 16,384, and choose regularization within the range evaluated by its authors.

From training pairs to generated queries. TIDE is distilled exclusively on within-batch pairs of *training* samples, whereas attribution queries are *generated* samples, introducing a potential train–query distribution shift. Three observations suggest that this shift does not substantially impair performance in our experiments. First, all counterfactual evaluations already test TIDE under this shift: the queries in Table 1 and Fig. 2 are generated samples, on which TIDE achieves attribution quality close to that of its teache. Second, we directly evaluate student–teacher agreement on generated queries using the deployed inference protocol (M=32 aligned draws, rankings over all training samples, 20 queries, and three runs per dataset). The student's rankings achieve Spearman correlations of 0.87, 0.89, and 0.86 with the teacher's on CIFAR-10, ArtBench-10, and MS-COCO, respectively, demonstrating that the learned embeddings transfer to generated queries. Third, embeddings are computed from *noised* states, where forward noising can reduce differences between the training and generated distributions. This provides a potential mechanism for the observed transfer from training to generation distribution.

Table 5: Attribution settings for TID and TIDE. We use the same configuration on all three datasets. Table 1 reports results with M=32, while Fig. 2 varies the budget over $\{1, 32, 100\}$.

		Setting
TID	Timestep schedule	Karras
	Corruption draw	shared
	Relative damping α	0.1
	Prenormalization	per sample
	Inference draws M	32
	Inference draw alignment	aligned $(\sigma, \mathbf{n})$
TIDE	Input features	residual stream
	Residual blocks used	12
	Residual combination	sum
	Pooling	attention
	Hidden width	512
	Embedding dimension	768
	Similarity head	cosine
	Scale $\alpha = e^{\rho}$, init.	32
	Distillation loss	LambdaRank
	Epochs	50
	Batch size	128
	Learning rate	5×10^{-5}
	Inference draws M	32
	Inference draw alignment	aligned $(\sigma, \mathbf{n})$

Table 6: Effective removal size, averaged over six seeds, with the fraction of the training set shown in parentheses. Each method selects the top 2% of samples for each of 20 queries; overlap among these per-query sets reduces the union below the nominal total of 40%. The random control has the least overlap and therefore removes the most distinct samples.

Method	CIFAR-10	ArtBench-10	MS-COCO
Random	16,636 (33.3%)	16,589 (33.2%)	39,332 (33.3%)
CLIP	13,025 (26.1%)	10,349 (20.7%)	34,384 (29.1%)
DINO	13,728 (27.5%)	11,857 (23.8%)	35,952 (30.4%)
D-TRAK	15,586 (31.2%)	13,874 (27.8%)	35,154 (29.7%)
DAS	13,758 (27.5%)	14,262 (28.6%)	38,006 (32.1%)
MUCS	15,448 (30.9%)	14,093 (28.2%)	34,096 (28.8%)
TID (ours)	15,419 (30.8%)	14,231 (28.5%)	37,467 (31.7%)
TIDE (ours)	15,886 (31.8%)	14,734 (29.5%)	36,942 (31.2%)

F Additional Results

F.1 Qualitative Analysis of Attribution Rankings

The counterfactual AUC measures the effect of removing selected samples, but does not reveal *which* samples each method selects. Figures 4–6 visualize the rankings for one generated query from each dataset. Each row corresponds to a method in Table 1 and shows, from left to right, the eight highest-ranked and eight lowest-ranked training samples (we omit the random baseline for clarity). All examples are taken directly from the deployed attribution outputs for the same query; no method is recomputed or re-tuned for this visualization.

The methods differ relatively little at the top of their rankings. For the MS-COCO skateboarding query, all methods retrieve skateboarding scenes. For the CIFAR-10 automobile query, all methods except D-TRAK primarily retrieve automobiles, whereas D-TRAK also selects watercraft, aircraft, and a dog. Because the model-agnostic encoders rank samples using visual similarity alone yet still produce semantically plausible top-ranked examples, agreement at the head of the ranking is not by itself evidence of faithful attribution. This observation is consistent with the relatively weak counterfactual performance of these encoders in Table 1.

Table 7: Average counterfactual AUC of TIDE on CIFAR-10 across per-query removal budgets. Performance increases consistently with the budget as larger removals induce more detectable changes in the retrained model.

	1%	2%	3%	5%
TIDE (ours)	0.784	0.858	0.910	0.938

The tails of the rankings reveal clearer differences. The encoder baselines tend to select semantically dissimilar examples: CLIP retrieves text-heavy posters, while DINO retrieves animals for the automobile query and food displays for the skateboarding query. In contrast, the lowest-ranked samples under TID and TIDE tend to have weak visual signal, including sparse line drawings on ArtBench-10, isolated objects on white backgrounds on CIFAR-10, and dim nighttime scenes on MS-COCO. This pattern is compatible with a gradient-geometric notion of influence, under which samples can be weakly related to the query even when they are not semantic opposites. DAS exhibits a different pattern on CIFAR-10: its lowest-ranked examples still include cars, boats, and aircraft, suggesting that the two ends of its ranking are not separated by a simple semantic axis.

These single-query examples are intended only to illustrate the content of the rankings. They are consistent with the quantitative results but do not independently establish them.

F.2 Sensitivity to the Removal Budget

Our main evaluation removes the top 2% of training samples for each query. To assess the sensitivity to this choice, Table 7 evaluates TIDE on CIFAR-10 using per-query removal budgets of 1, 2, 3, and 5%. We see that performance improves consistently as the budget increases, indicating that the result is not specific to the default operating point. We also observe a possible trend towards plateau as the budget crosses 3%, which could suggest that, for CIFAR-10, influence does not propagate significantly over such percentage of training items.

F.3 Relative Magnitude of the Removal Effect

In addition to AUC, Serrà et al. (2026) report the *similarity ratio*: the percentage change in similarity between each query and its regeneration after removing the attributed samples. Whereas AUC measures how reliably a method can be distinguished from the random-removal control, the ratio measures the magnitude of the induced change. Table 8 reports this quantity for each metric, together with their average, using the same seeds as in Table 1 (to homogenize results across metrics, in Table 8 we use $1 - \text{LPIPS}$, mimicking a similarity metric). More negative values indicate larger removal-induced drift; the random-removal control yields approximately -2 to -3%.

The ratio results broadly agree with the AUC ranking in Table 1. TID induces the largest average drift on ArtBench-10 and MS-COCO, while TID, TIDE, and MUCS perform similarly on CIFAR-10. Because the two metrics capture different properties, this agreement serves as a consistency check rather than a redundant evaluation: a method may reliably separate from the random-removal control while inducing only a small absolute change in the regenerated samples.

F.4 Additional Efficiency Details

The complexity split behind Figure 2. The per-query costs in Figure 2 are determined by which part of each method's computation is query-independent and small enough to cache. We use the following notation. N is the number of training samples, Q the number of queries, and M the per-sample budget of timesteps or draws. c_{fwd} and c_{fb} denote the cost of one forward pass and of one forward–backward pass of the diffusion model F_θ, and c_{enc} the cost of one pass of an external encoder, $c_{\text{enc,fb}}$ the cost of one forward–backward pass of the encoder. Denote by E the number of distillation epochs, and by U the number of unlearning steps. For the layer-wise factors, d_l^{in} and d_l^{out} are the input and output dimensions of layer l, and we abbreviate $D_1 = \sum_l (d_l^{\text{in}} + d_l^{\text{out}})$, $D_2 = \sum_l \left((d_l^{\text{in}})^2 + (d_l^{\text{out}})^2\right)$, and $D_3 = \sum_l \left((d_l^{\text{in}})^3 + (d_l^{\text{out}})^3\right)$, the sizes of the uncompressed factors, of the K-FAC matrices, and the computational cost of their eigendecompositions. Storage

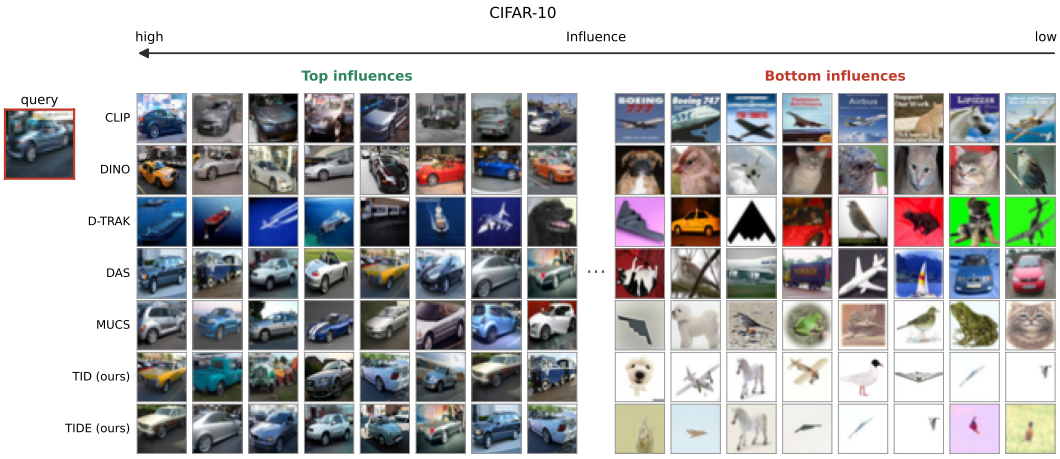


Figure 4: Qualitative rankings on CIFAR-10. Each row corresponds to a method in Table 1 and shows the query, the eight most influential training samples, and the eight least influential samples. The arrow indicates decreasing attributed influence.

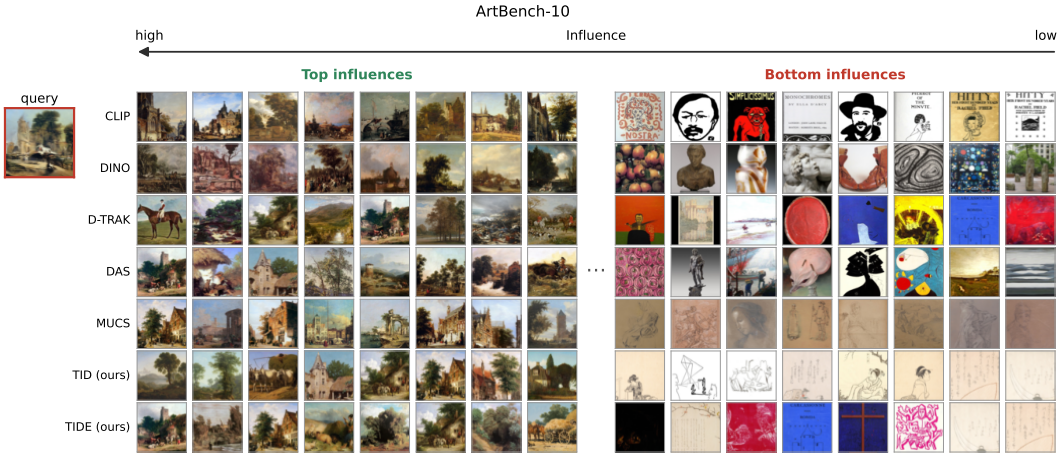


Figure 5: Qualitative rankings on ArtBench-10, following the format of Fig. 4. For this densely painted query, TID and TIDE place sparse line drawings at the bottom of their rankings.

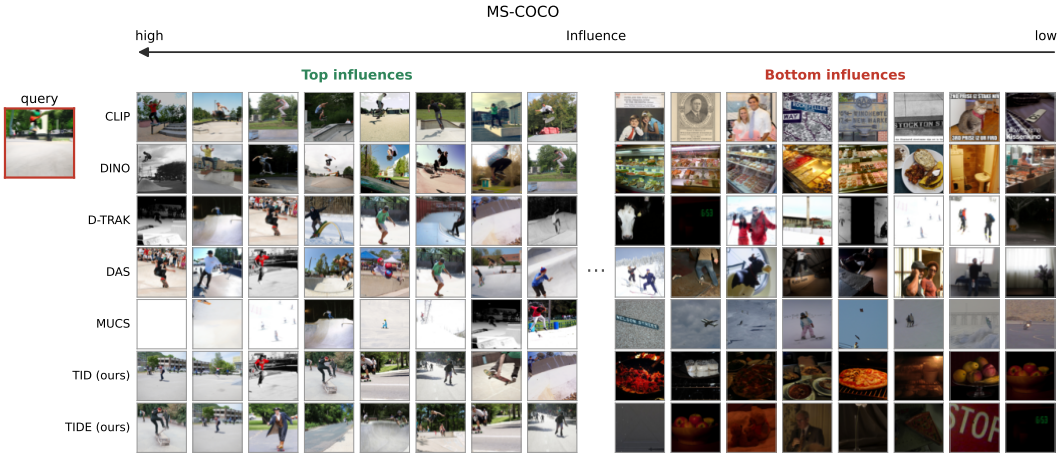


Figure 6: Qualitative rankings on MS-COCO, following the format of Fig. 4. All methods retrieve skateboarding scenes at the top of the ranking, while their lowest-ranked samples differ.

Table 8: Relative change in query–regeneration similarity (%) after removal, reported as the mean ± standard error across seeds. More negative values indicate larger induced drift. The random-removal control yields approximately −2 to −3%. Method categories follow Table 1; the strongest and second-strongest results in each column are shown in **bold** and underlined, respectively.

	Approach	Category	SSIM	SSCD	LPIPS	CLIP	μ
CIFAR-10	CLIP	MA	−3.9±0.6	−6.0±1.1	−5.1±0.7	−7.5±0.6	−5.6±0.7
	DINO	MA	−4.5±0.7	−7.5±0.8	−6.2±0.6	−8.0±0.9	−6.5±0.7
	D-TRAK	G	−6.6±0.8	−8.6±1.1	−8.2±0.8	−7.1±1.5	−7.6±1.0
	DAS	G	−9.2±1.2	−14.4±1.9	−12.0±1.4	−11.8±1.0	−11.9±1.3
	MUCS	U	−12.4±0.7	−19.2±0.9	−16.8±0.5	**−16.9**±0.9	**−16.3**±0.6
	TID (ours)	G	−12.7±0.8	−18.1±1.0	−16.9±0.5	−15.7±1.1	−15.8±0.8
	TIDE (ours)	D	**−13.2**±0.4	**−19.4**±0.5	**−17.5**±0.8	−15.0±1.1	**−16.3**±0.6
ArtBench-10	CLIP	MA	−2.4±1.1	−5.7±2.2	−3.6±1.8	−4.8±1.2	−4.1±1.5
	DINO	MA	−2.5±0.8	−5.3±1.8	−3.5±0.9	−3.7±1.2	−3.7±1.1
	D-TRAK	G	−8.7±1.7	−14.3±3.0	−9.7±2.4	−6.6±1.6	−9.8±2.2
	DAS	G	−11.2±1.5	−20.7±1.8	−13.4±2.0	−9.7±1.2	−13.7±1.6
	MUCS	U	−13.7±1.2	−23.9±1.5	−16.3±1.6	−11.9±1.0	−16.4±1.2
	TID (ours)	G	**−16.4**±1.3	**−28.6**±1.9	**−19.2**±1.6	**−13.1**±0.8	**−19.3**±1.3
	TIDE (ours)	D	−10.9±1.1	−20.7±2.1	−13.6±1.6	−9.6±1.2	−13.7±1.4
MS-COCO	CLIP	MA	−10.1±1.1	−19.5±3.5	−11.3±1.5	−11.9±0.9	−13.2±1.6
	DINO	MA	−11.0±1.2	−23.0±4.0	−12.7±1.4	**−12.1**±0.8	−14.7±1.7
	D-TRAK	IF	−12.6±1.1	−19.7±3.4	−13.9±1.5	−6.8±0.7	−13.2±1.3
	DAS	IF	−12.7±1.2	−22.5±4.0	−14.7±1.4	−8.1±0.9	−14.5±1.7
	MUCS	U	−13.3±1.0	−23.6±2.6	−15.3±1.3	−9.4±1.2	−15.4±1.2
	TID (ours)	IF	**−17.6**±1.3	**−28.8**±3.9	**−19.8**±1.1	−11.3±1.1	**−19.4**±1.7
	TIDE (ours)	D	−14.8±1.7	−24.6±4.1	−16.6±1.5	−9.1±0.7	−16.3±1.8

is counted in floats. The total model parameters $P = \sum_l \left(d_l^{\text{in}} d_l^{\text{out}}\right)$, and the embedding dimension d_{emb} and projection dimension d_{proj} satisfy $d_{\text{emb}} \ll d_{\text{proj}} \ll P$. Writing C_{total} for the total cost of attributing all Q queries and C_{cached} for its query-independent part, which is computed once and stored, the warm per-query cost plotted in Figure 2 is

$$c_{\text{query}} = \frac{C_{\text{total}} - C_{\text{cached}}}{Q}. \tag{21}$$

Table 9 lists both terms per method, separating the flops of the final score products, which are matrix multiplies rather than model evaluations. Below we derive the entries method by method:

- CLIP and DINO — The encoder embeds each training sample once, at cost Nc_{enc}, giving an Nd_{emb} store. Attributing a query costs one encoder pass and $N \times d_{\text{emb}}$ for inner products.
- D-TRAK and DAS — Each training gradient is projected to d_{proj} dimensions and averaged over M timesteps, so the train side reduces to an Nd_{proj} store computed once at cost NMc_{fb} (DAS additionally spends NMc_{fwd} on its normalization statistics). A query then costs its own M projected gradients, Mc_{fb}, and one $N \times d_{\text{proj}}$ vector product. This caching is what random projection buys, at the accuracy cost visible in Table 1.
- TID — The score in Eq. 5 pairs training and query factors from matching draws before averaging, so it cannot be computed from separately pre-averaged factors, and storing the per-draw factors themselves would take NMD_1 floats, on the order of a petabyte for CIFAR-10 at $M = 32$. Only the K-FAC factors and their eigendecompositions are cacheable, D_2 floats or a few GB, computed once at cost $NMc_{\text{fb}} + O(D_3)$. The NM train-side evaluations therefore recur on every pass, and N enters the per-query cost through Eq. 21 as $(NM/Q)\, c_{\text{fb}}$, in addition to the query's own Mc_{fb} and an $O(NMD_1)$ score product.
- TIDE — Distillation moves all gradient computation into the one-off phase: $O(EN)\,(c_{\text{fb}} + c_{\text{enc,fb}})$ to train the embedder, then $NM(c_{\text{fwd}} + c_{\text{enc}})$ to embed the training set into an NMd_{emb} store. A query costs $M(c_{\text{fwd}} + c_{\text{enc}})$ and one $N \times M \times d_{\text{emb}}$ vector product, the same warm shape as the encoder baselines.

Table 9: Computation and storage by phase. "One-off" is computed once on the training set and cached; "per query" is Eq. 21. c_{fwd}, c_{fb}, and c_{enc} are the costs of one forward pass, one forward–backward pass of F_θ, and one external-encoder pass; D_1, D_2 are the sizes of the uncompressed factors, the K-FAC matrices, and D_3 is the computational cost of their eigendecompositions. Cache sizes are in floats; score flops are the matrix multiplies of the final scoring step. We omit the big O notation from the whole table.

Method	One-off compute	Cache	Per-query compute	Score flops
CLIP / DINO	Nc_{enc}	Nd_{emb}	c_{enc}	Nd_{emb}
D-TRAK	NMc_{fb}	Nd_{proj}	Mc_{fb}	Nd_{proj}
DAS	$NMc_{\text{fwd}} + NMc_{\text{fb}}$	Nd_{proj}	Mc_{fb}	Nd_{proj}
TID	$NMc_{\text{fb}} + D_3$	D_2	$\frac{NM}{Q}c_{\text{fb}} + Mc_{\text{fb}}$	NMD_1
TIDE	$EN\,(c_{\text{fb}} + c_{\text{enc,fb}}) + NM(c_{\text{fwd}} + c_{\text{enc}})$	NMd_{emb}	$M(c_{\text{fwd}} + c_{\text{enc}})$	NMd_{emb}
MUCS	—	—	$Uc_{\text{fb}} + NMc_{\text{fwd}}$	NM

Table 10: One-off training-side cost and resulting cache size on a single H100. Encoder and embedding-bank sizes are computed as $N_{\text{train}}d \times 4$ bytes; all other cache sizes are measured on disk. The reported wall-clock times are direct measurements from a dedicated timing seed. "n/a" indicates that MUCS performs no training-side preprocessing.

Method	One-off computation	CIFAR-10	ArtBench-10	MS-COCO
		wall-clock / cache size		
CLIP	encode training set	98 s / 0.10 GB	112 s / 0.10 GB	277 s / 0.24 GB
DINO	encode training set	18 s / 0.15 GB	18 s / 0.15 GB	37 s / 0.36 GB
DAS	error + projected-gradient store	18.1 h / 3.3 GB	12.3 h / 3.3 GB	38.1 h / 7.8 GB
TIDE	distillation + embedding collection	3.8 h / 0.15 GB	4.2 h / 0.15 GB	10.0 h / 0.36 GB
TID (M=1)	factor estimation	0.2 h / 2.4 GB	0.2 h / 2.4 GB	0.5 h / 2.5 GB
TID (M=32)	factor estimation	1.7 h / 2.4 GB	2.7 h / 2.4 GB	6.4 h / 2.5 GB
TID (M=50)	factor estimation	2.6 h / 2.4 GB	4.3 h / 2.4 GB	10.0 h / 2.5 GB
TID (M=100)	factor estimation	5.0 h / 2.4 GB	8.2 h / 2.4 GB	20.1 h / 2.5 GB
MUCS	none	n/a	n/a	n/a

- *MUCS* — No computation is query-independent: each query fine-tunes the model for U steps, Uc_{fb}, and then sweeps the training set with NM forward passes, NMc_{fwd}, to measure loss changes, so both U and NM appear in the per-query cost and nothing is cached.

Training-side (one-off) costs. The warm-cache per-query costs in Figure 2 exclude the computation performed once on the training set and subsequently reused. Table 10 below reports these one-off costs and the resulting cache sizes for all three datasets on a single H100. For cacheable methods, preprocessing takes less than five minutes for the model-agnostic encoders, 3.8–10.0 h for TIDE, and 12.3–38.1 h for DAS. The TIDE cost is dominated by distillation; constructing the embedding bank itself requires only minutes. After preprocessing, these methods incur only the per-query costs reported in Figure 2.

For TID, factor estimation and eigendecomposition are cacheable, but the training gradients required by each attribution pass are not feasible to store at full scale and must therefore be recomputed (Sec. 3.3). MUCS performs no training-side preprocessing and incurs its full computation for every query. The one-off cost also grows faster than the number of training samples alone would predict, because higher-resolution inputs raise the per-sample cost. DAS evaluates $M = 100$ timesteps on CIFAR-10 and $M = 50$ on the two larger datasets, so normalizing its one-off cost by $|\mathcal{Z}| \cdot M$ isolates the cost of a single image–timestep, which is 13.0, 17.7, and 23.2 ms on CIFAR-10, ArtBench-10, and MS-COCO, respectively. Scaling the CIFAR-10 measurement by dataset size and timestep count alone would therefore predict 21.4 h on MS-COCO, well below the measured 38.1 h; the shortfall is the higher per-node cost of 64×64 inputs with CLIP conditioning.