Training Data Attribution in Diffusion Models via Mirrored Unlearning and Noise-Consistent Skew

Joan Serrà[1], **Dipam Goswami**[1*], **Fabio Morreale**[1], **Wei-Hsiang Liao**[1], **and Yuki Mitsufuji**[1,2]

[1] Sony AI
[2] Sony Group Corporation
name.surname@sony.com

Abstract

Training data attribution (TDA) should enable generative model interpretability and foster a variety of related downstream tasks. Nonetheless, current TDA approaches lack reliability and robustness, preventing their adoption in real-world setups. In this paper, we take a decisive step towards more reliable and robust TDA for diffusion models. We propose to perform TDA with mirrored unlearning and noise-consistent skew (MUCS). The idea is to fine-tune a second model with bounded mirrored gradient ascent, and to measure the normalized skew of this model with respect to the original one using consistent noise samples. We show that, while being conceptually simple and generic, MUCS systematically outperforms existing methods on three different datasets by a large margin. We additionally study the effect that core design choices have on final performance, and analyze novel aspects regarding the overlap of influential instances across generated items and the potential of ensembling TDA approaches. We believe that our findings may have broader implications for more general unlearning setups, as well as for tasks requiring the comparison of diffusion losses.

1 Introduction

In generative modeling, training data attribution (TDA) [1, 2] seeks to identify the training instances that causally influenced the generation of an item. Intuitively, such instances should be influential in a counterfactual sense [2–5], that is, removing them from training should prevent the model from generating the same item. This counterfactual notion of attribution is key to generative model interpretability [2], as well as to a myriad of downstream applications like data selection, leakage estimation, or memorization analysis [6–8]. Furthermore, there are domains in which attribution has direct practical value. For instance, in the music domain, TDA algorithms could inform the distribution of royalties derived from the generated samples [9–11]. All these applications, however, still lack widespread adoption due to the limited performance and poor generalization of current TDA algorithms. Besides, proper counterfactual evaluation is also computationally demanding [2, 8].

Diffusion models [12] are the most prominent generative paradigm in domains like images or audio. This has motivated the adaptation of previous TDA algorithms to such paradigm [3–5]. However, with the introduction of more sophisticated diffusion variants [12], the adaptation of those TDA algorithms can become difficult or infeasible. On the other hand, there are a few non-parametric or model-agnostic methods, typically leveraging some form of embedding-based similarity [6, 13], but those can fall short in terms of performance and do not embody any notion of causality (see also the baselines in, e.g., [3–5]). Unlearning is a relatively recent and general approach that has shown promising

[*] Work done during an internship at Sony AI.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

results for diffusion models, especially since the introduction of mirrored unlearning[1] [7, 8, 14]. Its core idea is that unlearning a generated item should measurably affect the model's performance on the influential training instances. However, existing unlearning-based approaches rely on approximations, involve sensitive hyperparameters, offer limited control over the unlearning process, and employ simplistic attribution-scoring methods.

In this work, we introduce a method for TDA that combines a redefined mirrored unlearning process together with improvements in measuring the effect induced by such process. In contrast to existing works, our mirrored unlearning process combines a retention and a forgetting loss, and adds control by defining a null performance target (Sec. 2.2) that is used both for preventing unnecessary unlearning and for stopping the unlearning process (Sec. 2.3). Our method also significantly differs from existing works in the way it performs attribution scoring (Sec. 2.4): instead of relying on simple subtractions between loss expectations using the training noise distribution, it averages individual noise-consistent normalized loss skews using the generation noise distribution. We not only show that these components independently yield important performance improvements, but also that the method as a whole clearly outperforms existing TDA approaches (Sec. 5). We also conduct an analysis of attributed item overlap, and study the performance of attribution ensembles. Overall, our work introduces a number of important findings for TDA, both in terms of mirrored unlearning and attribution scoring.

2 Method

2.1 Notation and Overview

Our method targets diffusion models trained under common principles [12], and is not specifically tied to individual network architectures, loss functions, diffusion variants, noise schedules, sampling schemes, or data types. As such, it only assumes a diffusion model F_1, pre-trained using some reconstruction loss L leveraging noise scales σ, noise vectors $\mathbf{n}$, and training data instances $\mathbf{z}$. The latter are formed by data items $\mathbf{x}$ and, possibly, conditioning signals $\mathbf{c}$, such that $\mathbf{z} = \{\mathbf{x}, \mathbf{c}\}$. We denote the training dataset by $\mathcal{Z} = \{\mathbf{z}_1, \dots \mathbf{z}_{|\mathcal{Z}|}\}$. At inference time, F_1 generates an item $\hat{\mathbf{x}}$ using conditioning $\hat{\mathbf{c}}$, forming $\hat{\mathbf{z}} = \{\hat{\mathbf{x}}, \hat{\mathbf{c}}\}$. For the sake of generalization, we make L implement the diffusion variant. For example, for DDPM [15],

$$L(\mathbf{z}, \sigma, \mathbf{n}, F) = \mathop{\mathbb{E}}_{\sigma,\mathbf{n},\mathbf{z}} \left[\left\| \mathbf{n} - F\left(\sqrt{\alpha(\sigma)}\,\mathbf{x} + \sqrt{1-\alpha(\sigma)}\,\mathbf{n}, \sigma, \mathbf{c}\right) \right\|^2 \right]$$

and, for EDM [16],

$$L(\mathbf{z}, \sigma, \mathbf{n}, F) = \mathop{\mathbb{E}}_{\sigma,\mathbf{n},\mathbf{z}} \left[\omega(\sigma) \left\| \mathbf{x} - F\left(\mathbf{x} + \sigma\mathbf{n}, \sigma, \mathbf{c}\right) \right\|^2 \right],$$

where α and ω are weighting functions that depend on σ (see [12] for further diffusion variants).

Given F_1, L, and $\hat{\mathbf{z}}$, our method computes an attribution score a_i for each training instance $\mathbf{z}_i \in \mathcal{Z}$, reflecting the degree to which $\mathbf{z}_i$ has contributed to (or been influential in) generating $\hat{\mathbf{x}}$:

$$a_i = A(\mathbf{z}_i | \hat{\mathbf{z}}, L, F_1),$$

where A denotes the TDA algorithm. The computation of a involves two equally-important steps: unlearning $\hat{\mathbf{z}}$ to obtain an unlearned model F_2 (Sec. 2.3) and averaging noise-consistent loss skews to obtain the final score a (Sec. 2.4). We term our method MUCS, for mirrored unlearning and noise-consistent loss skew. A one-page pseudo-code of the full method is provided in Appendix A.1. The code we used to run our experiments is released at `https://github.com/sony/mucs`.

2.2 Null Loss Reference

Before running MUCS, we estimate a null loss value which will serve as an upper bound reference for the random (or null) performance of F. For that, we use the randomly-initialized model weights, which we denote by F_0, and estimate

$$L_{\text{null}} = \mathop{\mathbb{E}}_{\sigma,\mathbf{n},\mathbf{z}} \left[L(\mathbf{z}, \sigma, \mathbf{n}, F_0) \right], \tag{1}$$

[1] Also called the "mirrored influence hypothesis" [7] or "unlearning the synthesized image" [8].

where σ and $\mathbf{n}$ are drawn from the same distributions used in the training of F_1, and $\mathbf{z}$ is sampled uniformly without replacement from $\mathcal{Z}$. In Eq. 1, the expectation can be computed from a few thousands of realizations or until we run out of data. We note that, alternatively, given normalized training data, L_{null} can be analytically derived for some common diffusion loss functions[2] [12].

2.3 Unlearning Approach

Given a generated item, MUCS proceeds by unlearning it as if it was part of the training set. This corresponds to the so-called mirrored unlearning paradigm [7, 8, 14], which assumes that unlearning a generated item $\hat{\mathbf{z}}$ will impact the model's performance on the training instances $\mathbf{z}_i$ that contributed to such generation. While this does not establish a causal relation from $\mathbf{z}_i$ to $\hat{\mathbf{z}}$, Ko et al. [7] and Wang et al. [8] empirically show that it significantly correlates with the impact that unlearning $\mathbf{z}_i$ has on $\hat{\mathbf{z}}$. Such correlation and impact can be measured directly using L, and exploited for TDA and related tasks (Sec. 1). Instead of performing direct gradient ascent on $\hat{\mathbf{z}}$ [7], or constraining it with the Fisher information matrix [8, 14], we propose to employ gradient ascent as a regularization to fine-tuning with training data.

Following mirrored unlearning, we aim to obtain a model F_2 that has unlearned the generated sample $\hat{\mathbf{z}}$. To do so, we copy the pre-trained model F_1 and perform fine-tuning (FT) with the train set $\mathcal{Z}$ and, at the same time, mirrored gradient ascent (GA) using $\hat{\mathbf{z}}$. Specifically, we iteratively minimize

$$\begin{aligned} L_{\text{MUCS}}\left(\mathcal{Z}, \hat{\mathbf{z}}, L, F_2\right) &= L_{\text{FT}}\left(\mathcal{Z}, L, F_2\right) - \lambda\, L_{\text{GA}}\left(\hat{\mathbf{z}}, L, F_2\right) \\ &= \mathop{\mathbb{E}}_{\sigma,\mathbf{n},\mathbf{z}}\left[\, L\left(\mathbf{z}, \sigma, \mathbf{n}, F_2\right)\right] - \lambda \mathop{\mathbb{E}}_{\sigma,\mathbf{n}}\left[\, \min\left(L\left(\hat{\mathbf{z}}, \sigma, \mathbf{n}, F_2\right), L_{\text{null}}\right)\right], \end{aligned} \tag{2}$$

where σ and $\mathbf{n}$ are drawn from the same distributions used in the training of F_1, $\mathbf{z}$ is sampled uniformly without replacement from $\mathcal{Z}$, and $\lambda > 0$ is a hyperparameter that controls the amount of unlearning (notice the negative sign). In Eq. 2, expectations are estimated from batches containing 100 independent draws of σ and $\mathbf{n}$ (and $\mathbf{z}$). In preliminary experiments, we observed that performance was largely insensitive to the choice of batch size.

Considerations — Having defined L_{MUCS}, we now make a number of considerations for better unlearning, and later validate them on the attribution task with an ablation study (Sec. 5).

- C1. First of all, note that the joint optimization of L_{FT} and L_{GA} will encourage a balanced unlearning step where training data information is preserved but, at the same time, information related to $\hat{\mathbf{z}}$ is destroyed. The hyperparameter λ will control such balance, and we typically want to constrain it to the range $\lambda \in (0, 1]$ to prevent an overly aggressive unlearning. After some preliminary experiments, we set $\lambda = 0.2$.
- C2. One important consideration for unlearning algorithms is how to control the amount of unlearning, that is, how to produce a meaningful unlearning effect while not destroying the overall generation performance. Here, we employ a twofold strategy: we control the amount of unlearning by clamping to L_{null} (Eq. 2) and stop updating the model once L_{GA} reaches a value close to random or null performance (e.g., $L_{\text{GA}} \geq 0.95 \cdot L_{\text{null}}$). Using the former prevents pushing the unlearned model towards extreme loss values, avoiding unrealistic unlearning. Using the latter allows for more dynamic unlearning processes which depend on how difficult it is to push the generated sample $\hat{\mathbf{z}}$ towards null model performance, avoiding insufficient or excessive unlearning steps on a case-by-case basis.
- C3. We believe that employing the same configuration as in pre-training is also important for robust unlearning. Hence, to minimize L_{MUCS}, we employ the same noise schedule, loss function, optimizer type, and hyperparameters used in the pre-training of F_1. The only exception to the above rule is the learning rate, which can actually vary during pre-training. By default, we set it constant and one order of magnitude below the nominal pre-training value.
- C4. Finally, as it has been reported that unlearning only certain parts of the model can yield some improvement for TDA [8], we perform unlearning only on the MLP weights of F_2 (both in the Transformer and in the conditioning network, see Appendix A.3). Although in our case we do not expect a huge difference in performance, unlearning only certain blocks

[2] For example, given image data between -1 and 1, the L_{null} for a well-calibrated EDM approach [16] is 1.

or certain parts of the model is customary in fine-tuning tasks [17], as well as in generic unlearning setups [18].

2.4 Attribution Score

With the unlearned model F_2, we can now proceed to attribute (or score the influence of) the training instances $\mathbf{z}_i$. To do so, we first form a set of pairs of noise scales σ and noise vectors $\mathbf{n}$, $\mathcal{N} = \{(\sigma_1, \mathbf{n}_1), \dots (\sigma_{|\mathcal{N}|}, \mathbf{n}_{|\mathcal{N}|})\}$. We do it by taking σ from the schedule used during generation and drawing $\mathbf{n}$ from the same distribution used during training. Then, for each training instance $\mathbf{z}_i \in \mathcal{Z}$, we compute the normalized skew between the unlearned and the original model losses, and average across pairs:

$$A_{\text{MUCS}}(\mathbf{z}_i|\hat{\mathbf{z}}, L, F_1) = \frac{1}{|\mathcal{N}|} \sum_{(\sigma_j, \mathbf{n}_j) \in \mathcal{N}} \frac{L\left(\mathbf{z}_i, \sigma_j, \mathbf{n}_j, F_2\right) - L\left(\mathbf{z}_i, \sigma_j, \mathbf{n}_j, F_1\right)}{\left|L\left(\mathbf{z}_i, \sigma_j, \mathbf{n}_j, F_2\right)\right| + \left|L\left(\mathbf{z}_i, \sigma_j, \mathbf{n}_j, F_1\right)\right| + \epsilon}, \tag{3}$$

where $\epsilon > 0$ is a small constant added for numerical stability (e.g., $\epsilon = 10^{-3}$). We pre-sample $|\mathcal{N}| = 100$ pairs and, importantly, we reuse the same set $\mathcal{N}$ for every $\mathbf{z}_i \in \mathcal{Z}$. In preliminary experiments, we observed that increasing the size of $\mathcal{N}$ slightly improves performance, but incurs extra computational cost.

Considerations — Having defined A_{MUCS}, we now make a number of considerations with regard to the novelty of its components, and later assess their impact with our ablation study (Sec. 5).

- C1. First of all, we note that A_{MUCS} utilizes a normalized skew formula, which differs from the common loss subtractions or dot products used in existing attribution approaches [1, 2], including the unlearning-based ones [7, 8, 14]. We advocate for using normalized skews instead of subtractions or dot products in order to compensate for abnormally high or low losses $L_i = L(\mathbf{z}_i, \sigma, \mathbf{n}, F_1)$. A normalized skew can be interpreted as a symmetric ratio[3] or a normalized subtraction, so that it produces a different effect depending on L_i (e.g., it is different to have a loss difference of 0.1 if $L_i = 0.1$ than if $L_i = 0.5$). This way, any loss change induced by unlearning $\hat{\mathbf{z}}$ is measured in relation to the original loss.
- C2. Another consideration to make, and which also differs from existing TDA approaches, is the fact that we use noise-consistent measurements for computing A_{MUCS}. More specifically, we compute all the loss terms in Eq. 3 using the same gain value σ_j and noise realization $\mathbf{n}_j$, and always calculate the average over the same set of elements $\mathcal{N}$. That is, we employ the same $(\sigma_j, \mathbf{n}_j)$ in all operations, and the same $\mathcal{N}$ for every $\mathbf{z}_i \in \mathcal{Z}$. Without noise-consistency, the quantities in Eq. 3 could be dominated by random fluctuations due to different noise realizations, as diffusion loss values can vary a lot with different σ and, more importantly, with different $\mathbf{n}$.
- C3. Finally, we find it relevant for A_{MUCS} to take the noise scales σ from the schedule used during the generation of $\hat{\mathbf{x}}$, instead of drawing them from the distribution used during the training of F_1 (compare with L_{MUCS} in Eq. 2). In the image domain, it is well-known that large noise scales are more influential for general item properties such as object placement, spatial distribution, and overall color choices (see, e.g., [19]). By employing the generation schedule instead of the training distribution, which usually concentrates more on intermediate noise scales (see, e.g., [16]), we prime A_{MUCS} to focus more on those general properties. In addition, since it is generally observed that the last diffusion steps (lowest σ) are only responsible for small details and textures (see, e.g., [3]), we decide to discard the last 30% of the schedule. Specifically, we obtain $\lfloor 100/0.7 \rfloor$ values for σ following the generation schedule (in descending order), and include only the first 100 in $\mathcal{N}$.

3 Related Works

There are only a few works performing TDA in diffusion models via unlearning. In principle, unlearning every item in the training set is not practical due to the size of existing datasets, but the work of Ko et al. [7] shifts this perspective with the mirrored influence approach. They perform unlearning with simple gradient ascent on the synthesized image. Concurrently, Wang et al. [8]

[3] We also experimented with plain loss ratios, obtaining slightly worse and less numerically-stable results.

also proposed to unlearn the synthesized image through gradient ascent. They introduce the Fisher information matrix as a constraint to prevent the unlearning of important weights, a technique borrowed from elastic weight consolidation (EWC) [20]. Choi et al. [14] adapt the latter approach to the music domain and on a large scale. None of these approaches includes any fine-tuning notion nor employ loss skews. Moreover, they perform unbounded unlearning using a constant number of steps (or gradient accumulations), and compute loss subtractions using non-consistent noise realizations.

Besides unlearning, there are other strategies for TDA in diffusion models. The most common (albeit computationally demanding) one is to consider the notion of influence functions [21], and to develop it for specific diffusion variants. Georgiev et al. [3] and Zheng et al. [4] do that for DDPM, incorporating the use of random projections to make gradient and Hessian calculations tractable, following TRAK [22]. Lin et al. [5] go one step further and introduce vector normalization and the KL divergence into the scoring procedure, obtaining state-of-the-art results. There are also several approaches that do not exploit the notion of influence functions (e.g., [23–25]), and some recent methods that attempt to perform fast, embedding-based attribution (e.g., [6, 13, 26]). The latter may however be model- and/or domain-specific. For a general introduction to TDA, we refer the reader to the surveys by Hammoudeh and Lowd [1] and Deng et al. [2]. To the best of our knowledge, previous TDA approaches do not consider the use of a normalized loss skew, noise-consistent realizations, and inference schedules for score calculation. Many TDA approaches also focus on specific diffusion variants like DDPM, which sometimes complicates their adaptation to more modern developments like EDM or flow matching [12].

To put our unlearning contribution into perspective, it is also worth to review existing unlearning approaches that are not tailored to TDA or diffusion models. Like ours, Alberti et al.'s approach [27] can be understood to perform both fine-tuning and gradient ascent on selected samples. Similarly, Heng and Soh [28] consider fine-tuning and gradient ascent for continual learning, but constrain the latter with EWC. Both of these works do not explicitly consider reusing the original model configuration, employ a constant number of unlearning steps, and do not leverage any notion of random performance or a null loss. In general, unlearning methods employ alternative optimization objectives. For example, Golatkar et al. [29] employ a KL divergence objective instead of plain gradient ascent, Grandikota et al. [30] use the predictions of the original (frozen) model instead of the available ground truth, Wu et al. [31] replace targets with randomly sampled instances, and Zhang et al. [32] and Shi et al. [33] employ strategies inspired by classifier-free guidance [34]. Additional approaches with similar considerations have been proposed for continual learning [35], preference optimization [36], classifiers [37, 38], or large language models [39, 40]. Overall, we could not find an unlearning approach that matches most of the aspects of our unlearning formulation.

4 Evaluation Methodology

Data — To demonstrate generalization with respect to different types of images, resolutions, and conditioning signals, we consider three complementary datasets: CIFAR10 [41], ArtBench10 [42], and the captioned version of MS-COCO [43], which we denote as COCO. For CIFAR10 we use a resolution of 32×32, while for ArtBench10 and COCO we use a resolution of 64×64. To simulate different scenarios, we do not employ any conditioning for CIFAR10, condition on the class index for ArtBench10, and use the CLIP-Text [44] embeddings of the original captions for COCO. For all datasets, we use the full original train/test splits. Although these datasets are routinely used in prior work, they are seldom considered jointly, and evaluations are often limited to a few classes (e.g., two classes from CIFAR10 or two classes from ArtBench10). Besides, very few TDA approaches consider conditional diffusion models. Although we are still far from the real-world datasets used to train current large-scale diffusion models, this setup substantially improves over existing approaches and pushes the TDA evaluation standards towards a more realistic scenario.

Diffusion Model — Our models follow a typical diffusion pipeline based on the DiT/B architecture [45] and the EDM variant [16]. For computational and practical reasons, the model works in the pixel space, and uses a two-layer convolutional encoder and decoder which are jointly trained with the main Transformer blocks. For conditional generation, we employ condition dropout during training and classifier-free guidance [34] during inference. Weights are initialized using PyTorch's [46] defaults, and we optimize them with AdamW [47]. In preliminary work, we conducted a few experiments with the DiT/L and UNet architectures, and also with the DDPM variant [15], obtaining equivalent results (not reported due to computational constraints). Appendix A.3 provides additional

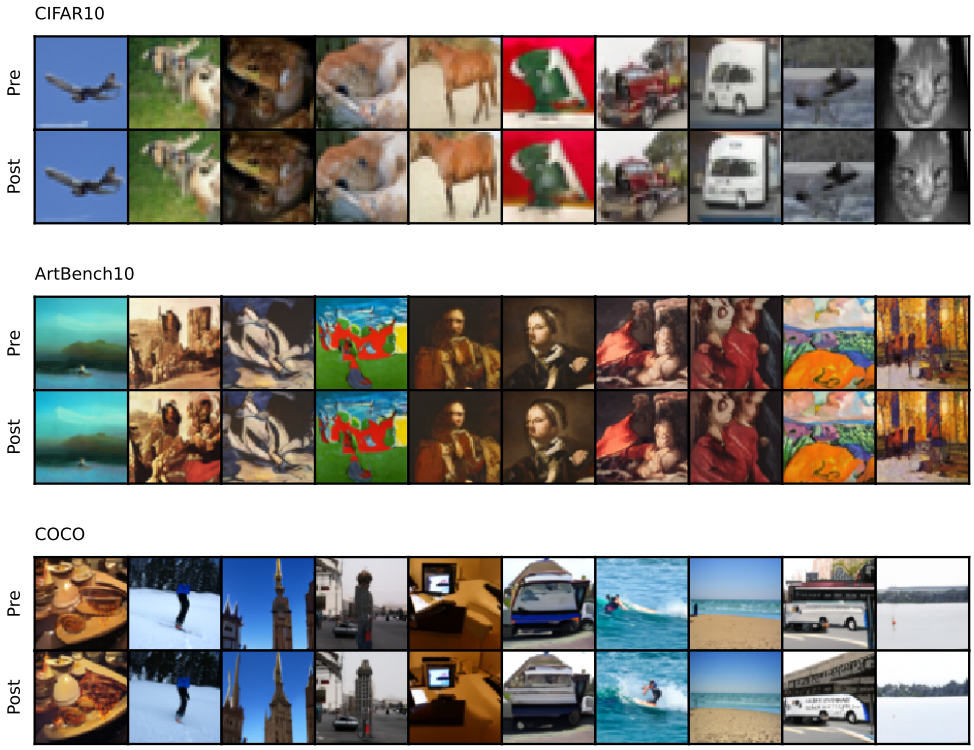

Figure 1: Examples of images generated using the original pre-trained model (Pre) and the post-trained model with 40% random data removal (Post). We can observe the strikingly high similarity for images generated with the same seed (columns), despite using a model retrained from scratch with a considerably smaller subset of the original data.

specifications of the training and generation procedures, together with the configuration differences for each dataset.

Baselines — To fairly compare with existing methods, we re-implement several baseline approaches, which we group under three broad categories: model-agnostic, influence function-based, and unlearning-based. Model-agnostic approaches rely on existing similarity measures to derive attribution scores. We consider cosine similarity of the condition vector (Condition; one-hot for CIFAR10/ArtBench10, CLIP-Text for COCO), cosine similarity in the CLIP-Image space (CLIP) [44], and cosine similarity in the DINOv2 space (DINO) [48]. Influence function-based approaches rely on gradient and Hessian calculations from multiple noise samples. We consider two of the most competitive approaches in this area: diffusion TRAK (D-TRAK) [4] and diffusion attribution score (DAS) [5]. For computational reasons, we project gradients to 16384 dimensions (preliminary experiments with a larger dimensionality were unstable and did not yield a dramatic improvement), and fine-tune the other hyperparameters on each dataset. Unlearning-based approaches leverage some techniques like the ones discussed in this paper. We consider the two existing approaches: the mirrored influence hypothesis approach (Forward-INF) [7] and the Hessian-weighted approach of attribution by unlearning (AbU) [8]. We fine-tune their hyperparameters on the CIFAR10 dataset. All approaches that consider batches of noise samples use 100 samples like ours.

Attribution Performance — We evaluate TDA using a leave-k-out counterfactual setup [2, 8] (see also [3–5]; considerations about alternative evaluation approaches are given in Appendix A.2). Essentially, this setup tests the ability of a model to generate a previously generated image after retraining from scratch without including its topmost k influential items in the training set. To quantify this ability, we rely on the seed-consistency property of diffusion models [3, 8], which implies that randomly removing non-influential items and retraining yields generations that are strikingly similar to the ones produced by the original model using the same seed (see Fig. 1 for examples, and also [3, 8]). To assess image similarity, we employ four measures covering multiple

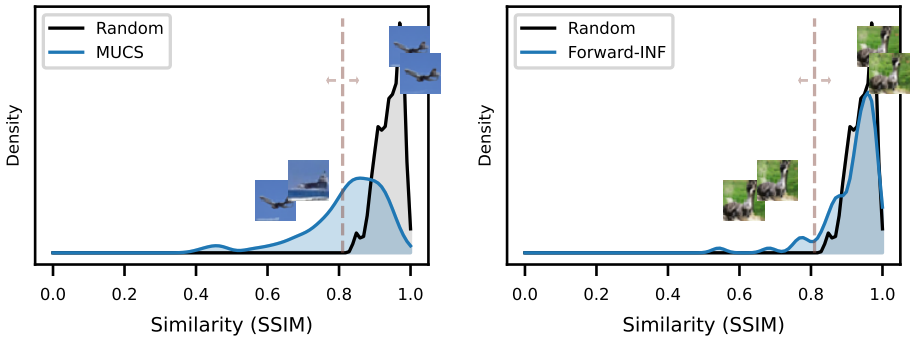

Figure 2: Examples of the distributions of similarities between $\mathbf{z}$ and $\mathbf{z}'$ for models trained on CIFAR10. Black and blue colors correspond to A_{rand} and a specific method A, respectively (see text). Better attribution algorithms (left) present less overlap than worse ones (right).

and complementary aspects: the structural similarity index measure (SSIM) [49], cosine similarity in the SSCD copy-detection space (SSCD) [50], the learned perceptual image patch similarity (LPIPS) measure [51], and cosine similarity in the semantic CLIP space (CLIP) [44].

The main idea behind our evaluation is that good TDA algorithms should select training items that, once removed from the training set, significantly alter the generations of the retrained model (compared to removing items at random, using the same seed). Given a set of m items $\hat{\mathcal{Z}} = \{\hat{\mathbf{z}}_1, \dots \hat{\mathbf{z}}_m\}$ generated using seeds $\mathcal{U} = \{u_1, \dots u_m\}$, we compute the top-$k$ most influential training items for each $\hat{\mathbf{z}}_i \in \hat{\mathcal{Z}}$ using some attribution method A. Next, we remove those items from the train set (mk removals in total) and retrain the diffusion model from scratch. We then generate a new set of items $\hat{\mathcal{Z}}' = \{\hat{\mathbf{z}}'_1, \dots \hat{\mathbf{z}}'_m\}$ using the same seeds $\mathcal{U}$, and compare their paired image similarities $\mathcal{S} = \{s_1, \dots s_m\}$, $s_i = S(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i)$, where S corresponds to the similarity metrics above. We perform this process for every attribution algorithm A and a random null model A_{rand} that simply attributes items at random.

To assess the degree of change produced by A with respect to A_{rand}, we measure the overlap between similarity distributions (Fig. 2). To objectively quantify such overlap, we use the area under the ROC curve (AUC), which is a common evaluation measure employed for that purpose [52]. The AUC also has the nice property of being proportional to the U statistic of the Mann-Whitney U test for measuring whether two samples correspond to the same distribution [53], and we leverage that to compute statistical significance (one tail, $p < 0.05$). Besides AUCs, we also report average similarity differences and their confidence intervals, so that we obtain a complementary perspective on distribution overlap (Appendix B.1). Our final setup involves removing $k = 2\%$ of the training data for $m = 20$ generations (totaling a maximum of 40% of the training data) and running it 6 times using different network and seed initializations. Hence, evaluations are made over 120 generated items, which is similar to or more than previous leave-k-out counterfactual evaluations (cf. [3–5, 8]). A more elaborate description of our evaluation approach can be found in Appendix A.2.

5 Results

We start by comparing MUCS with existing approaches (Table 1). We see that MUCS outperforms all considered approaches in all similarity metrics by a large margin. The only exception among the 12 considered cases is the CLIP metric in the COCO dataset. The difference with the best-performing baseline is statistically significant for almost all metrics in both CIFAR10 and ArtBench10 datasets. For COCO, the difference is not as significant, yet still substantial (COCO is a more complex dataset, and statistical significance is hard to obtain unless the baseline performs poorly). Besides MUCS, it is interesting to note that D-TRAK and DAS consistently perform better than model-agnostic or the other unlearning approaches. Model-agnostic approaches like CLIP or DINO can be very competitive in some datasets and metrics (e.g., CLIP in COCO). However, we see that their performance on other datasets can be relatively poor (e.g., CIFAR10 and ArtBench10). Previous unlearning-based methods do not perform well in general, perhaps due to their suboptimal formulation

Table 1: Comparison with existing approaches, which can be categorized into model-agnostic (MA), influence function-based (IF), and unlearning-based (U). We report AUC values for the considered similarity measures, plus their average (μ; last column). The asterisk (*) denotes statistical significance with respect to the closest baseline (Mann-Whitney U; $p < 0.05$).

	Approach	Categ.	SSIM	SSCD	LPIPS	CLIP	μ
CIFAR10	Condition	MA	0.572	0.576	0.568	0.550	0.566
	CLIP	MA	0.650	0.672	0.672	0.646	0.660
	DINO	MA	0.633	0.635	0.648	0.605	0.630
	D-TRAK [4]	IF	0.749	0.702	0.736	0.621	0.702
	DAS [5]	IF	0.810	0.787	0.813	0.735	0.786
	Forward-INF [7]	U	0.570	0.523	0.554	0.507	0.539
	AbU [8]	U	0.523	0.556	0.537	0.576	0.548
	MUCS (proposed)	U	**0.889***	**0.901***	**0.930***	**0.868***	**0.897**
ArtBench10	Condition	MA	0.547	0.542	0.547	0.560	0.549
	CLIP	MA	0.641	0.643	0.648	0.685	0.654
	DINO	MA	0.565	0.573	0.579	0.586	0.576
	D-TRAK [4]	IF	0.769	0.744	0.764	0.715	0.748
	DAS [5]	IF	0.783	0.762	0.782	0.722	0.762
	Forward-INF [7]	U	0.669	0.627	0.650	0.623	0.642
	AbU [8]	U	0.503	0.530	0.503	0.505	0.510
	MUCS (proposed)	U	**0.850**	**0.840***	**0.859***	**0.823***	**0.843**
COCO	Condition	MA	0.610	0.645	0.630	0.633	0.629
	CLIP	MA	0.678	0.705	0.719	0.764	0.716
	DINO	MA	0.682	0.720	0.710	**0.766**	0.720
	D-TRAK [4]	IF	0.720	0.708	0.742	0.683	0.713
	DAS [5]	IF	0.726	0.713	0.743	0.680	0.715
	Forward-INF [7]	U	0.616	0.620	0.634	0.624	0.624
	AbU [8]	U	0.531	0.523	0.543	0.526	0.531
	MUCS (proposed)	U	**0.750**	**0.763**	**0.799***	0.715	**0.757**

and the difficulty to setup some of their hyperparameters. MUCS overcomes this situation with significant improvement, while also being 1.5–2.9 times faster than the considered influence function- and unlearning-based approaches. This and additional results are available in Appendix B.1, together with image regeneration examples.

Next, we turn our attention to the specific MUCS components and their effect on overall performance (Table 2). We first observe that the hyperparameter λ has a mild effect when it is kept close to the chosen value of 0.2, with a global performance variation around 3% on COCO (U–C1). Sampling σ values from a slightly modified distribution during unlearning has a similar effect (U–C2), but we note that it can potentially be beneficial for some metrics (see CLIP; we leave this aspect open for future research). Employing the concept of L_{null} for both preventing excessive unlearning and deciding when to stop proves important, with a performance drop above 4% when using the MUCS average number of performed unlearning steps (U–C3). Unlearning the full Transformer block yields a performance drop around 5%. It is however in the part concerning attribution score calculation where our design choices have the most impact (Table 2, bottom). Substituting our normalized loss skew formulation by simple loss subtraction yields a performance drop of about 6% (S–C1), and using the training distribution for sampling σ causes another 5% drop (S–C3). Using non-consistent noises has the highest impact, with AUC scores dropping around 19% on average (S–C2). We hope that these design choices can be adopted by other TDA approaches beyond unlearning-based ones.

To conclude, we complement our work with an analysis of the overlap between top-k attributed instances and an exploration of TDA ensembles. In all our experiments, we observe that all considered approaches present some top-k overlap between generated items (Appendix B.2). Having an overlap implies that there are some training instances that are influential for multiple generations at the same time, and suggests the potential existence of some 'hub' or 'centrality' bias [54]. However, the overlaps observed with our experiments are relatively low, and close to random choice. While this deserves further investigation, we believe the observed degree of overlap is not critical for the evaluation of the considered approaches. If, instead of studying the top-k overlap obtained

Table 2: Ablation study on COCO. "Abl." refers to the MUCS step and consideration where the ablation is applied (e.g., U–C1 denotes the unlearning step, consideration 1; S–C2 refers to the scoring step, consideration 2). "Description" summarizes what changes with respect to the proposed approach (see also main text). Numbers correspond to AUC values for the considered similarity measures, plus their average relative difference (Δ; last column).

Abl.	Description	SSIM	SSCD	LPIPS	CLIP	Δ
	MUCS (proposed)	**0.750**	**0.763**	**0.799**	0.715	(0.757)
U–C1	Using $\lambda = 0.1$	0.733	0.737	0.767	0.695	-3.2%
U–C1	Using $\lambda = 0.3$	0.733	0.735	0.782	0.690	-2.9%
U–C2	Different σ distribution	0.734	0.732	0.771	**0.716**	-2.5%
U–C3	No L_{null} and 109 unlearning steps	0.716	0.713	0.764	0.707	-4.2%
U–C4	Unlearning full Transformer block	0.730	0.736	0.779	0.681	-3.3%
S–C1	Loss subtraction (no skew)	0.703	0.724	0.753	0.676	-5.7%
S–C2	Non-consistent noise pairs	0.618	0.621	0.619	0.596	-19.0%
S–C3	Full σ generation schedule	0.745	0.747	0.795	0.695	-1.6%
S–C3	Use of σ training distribution	0.709	0.708	0.761	0.692	-5.3%

Table 3: Ensemble results for the considered datasets. Only average AUC scores per dataset are reported (μ). Letters denote the MUCS reference (R) and the ensemble name (A–D).

#	Weight in the Ensemble							AUC μ		
	MUCS	DAS	D-TRAK	DINO	CLIP	Cond	F-INF	**CIFAR10**	**ArtBench10**	**COCO**
R	10	0	0	0	0	0	0	**0.897**	0.843	0.757
A	10	7	6	5	5	3	2	0.834	**0.859**	**0.801**
B	10	0	0	5	5	3	0	0.713	0.813	**0.801**
C	0	7	6	5	5	3	2	0.782	0.810	0.762
D	0	0	0	5	5	3	0	0.605	0.709	0.755

between generated items, we study the top-k overlap between attribution approaches, we observe that almost none of them presents an overlap over 35% with another approach (Appendix B.3). This fact suggests that perhaps an ensemble approach could provide a significant boost to the task of TDA (to our knowledge, such an ensemble approach has not been previously considered in the literature). To test this hypothesis, we implement and run a simple rank-based ensembling approach (Appendix A.4) and compute AUC values under different weightings (Table 3). Interestingly, we observe that ensembling has almost always a detrimental effect for datasets in which we already obtained good AUCs (e.g., CIFAR10 and ArtBench10), but can have a positive effect for datasets where AUCs were not so good (e.g., COCO). Of course, the cost of performing an ensemble increases with the cost of the individual methods being considered, but we observe that combining MUCS with cheap, model-agnostic attribution measures like DINO, CLIP, or Condition can provide a boost in some datasets (ensemble B, COCO). We also observe that removing MUCS from the ensemble has a notable detrimental effect (ensemble A vs. C and B vs. D). In future work, it would be interesting to study how and when to combine TDA approaches to form an ensemble, especially including computationally-cheap approaches like model-agnostic or non-parametric ones (Sec. 3).

6 Conclusion

This paper proposes core methodologies for performing TDA in diffusion models. On the one hand, we propose an improved mirrored unlearning approach based on fine-tuning and gradient ascent, featuring gradient ascent clamping and a dynamic stopping criterion. On the other hand, we propose several performance-critical changes to how attribution scores should be computed, including noise-consistency, normalized skews, and using part of the generation noise schedule. The combination of these core methodologies yields MUCS, a TDA approach that significantly outperforms existing approaches on counterfactual evaluations using three different datasets. We hope that the proposed methodologies can further impact more general unlearning setups and tasks where diffusion losses need to be compared.

Acknowledgements

We thank Woosung Choi for his AbU code review, and Chieh-hsin Lai and Yuhta Takida for initial discussions before starting this project.

References

[1] Z. Hammoudeh and D. Lowd. Training data influence analysis and estimation: a survey. *Machine Learning*, 113:2351–2403, 2024.

[2] J. Deng, Y. Hu, P. Hu, T.-W. Li, S. Liu, J. T. Wang, D. Ley, Q. Dai, B. Huang, J. Huang, C. Jiao, H. A. Just, Y. Pan, J. Shen, Y. Tu, W. Wang, X. Wang, S. Zhang, S. Zhang, R. Jia, H. Lakkaraju, H. Peng, W. Tang, C. Xiong, J. Zhao, H. Tong, H. Zhao, and Jiaqi W Ma. A survey of data attribution: methods, applications, and evaluation in the era of generative AI. *SSRN: 5451054*, 2025.

[3] K. Georgiev, J. Vendrow, H. Salman, S. M. Park, and A. Madry. The journey, not the destination: how data guides diffusion models. In *Proc. of the ICML Workshop on Challenges in Deployable Generative AI*, 2023.

[4] X. Zheng, T. Pang, C. Du, J. Jiang, and M. Lin. Intriguing properties of data attribution on diffusion models. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2024.

[5] J. Lin, L. Tao, M. Dong, and C. Xu. Diffusion attribution score: evaluating training data influence in diffusion models. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2024.

[6] W. Sun, H. Liu, N. Kandpal, C. Raffel, and Y. Yang. Enhancing training data attribution with representational optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, page in press. 2025.

[7] M. Ko, F. Kang, W. Shi, M. Jin, Z. Yu, and R. Jia. The mirrored influence hypothesis: efficient data influence estimation by harnessing forward passes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 26286–26295, 2024.

[8] S.-Y. Wang, A. Hertzmann, A. A. Efros, J.-Y. Zhu, and R. Zhang. Data attribution for text-to-image models by unlearning synthesized images. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 4235–4266. 2024.

[9] J. Deng and J. Ma. Computational copyright: towards a royalty model for music generative AI. In *ICLR Workshop on Navigating and Addressing Problems for Foundation Models (DPFM)*, 2024.

[10] F. Morreale, W. Hutiri, J. Serrà, A. Xiang, and Y. Mitsufuji. Attribution-by-design: ensuring inference-time provenance in generative music systems. *ArXiv: 2510.08062*, 2025.

[11] W. Kim, H. Wi, S. Park, T. Kim, S. Keum, K. Kim, T. Kim, J. Jung, T. Kim, G. Guerrero, M. Le Goff, J. Po, D. Moon, J. Nam, and J. Lee. From generation to attribution: music AI agent architectures for the post-streaming era. In *Proc. of the AI for Music Workshop at NeurIPS25 (AI4Music)*, 2025.

[12] C.-H. Lai, Y. Song, D. Kim, Y. Mitsufuji, and S. Ermon. The principles of diffusion models. *ArXiv: 2510.21890*, 2025.

[13] Y. Zhao, C. Du, X. Zheng, T. Pang, and M. Lin. Nonparametric data attribution for diffusion models. *ArXiv: 2510.14269*, 2025.

[14] W. Choi, J. Koo, K. Cheuk, J. Serrà, M. A. Martínez-Ramírez, Y. Ikemiya, N. Murata, Y. Takida, W.-H. Liao, and Y. Mitsufuji. Large-scale training data attribution for music generative models via unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Creative AI Track, page in press. 2025.

[15] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6840–6851. 2020.

[16] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 26565–26577. 2022.

[17] N. Kumari, B. Zhang, R. Zhang, E. Shechtman, and J.-Y. Zhu. Multi-concept customization of text-to-image diffusion. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1931–1941, 2023.

[18] N. Kumari, B. Zhang, S.-Y. Wang, E. Shechtman, R. Zhang, and J.-Y. Zhu. Ablating concepts in text-to-image diffusion models. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 22691–22702, 2023.

[19] Q. Bertrand, A. Gagneux, M. Massias, and R. Emonet. On the closed-form of flow matching: generalization does not arise from target stochasticity. In *Advances in Neural Information Processing Systems (NeurIPS)*, number in press. 2025.

[20] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, 2017.

[21] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 1885–1894, 2017.

[22] S. M. Park, K. Georgiev, A. Ilyas, G. Leclerc, and A. Madry. TRAK: attributing model behavior at scale. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 27074–27113, 2023.

[23] Zheng Dai and David K. Gifford. Ablation based counterfactuals. *ArXiv: 2406.07908*, 2024.

[24] J. Brokman, O. Hofman, R. Vainshtein, A. Giloni, T. Shimizu, I. Singh, O. Rachmil, A. Zolfi, A. Shabtai, Y. Unno, and H. Kojima. MONTRAGE: monitoring training for attribution of generative diffusion models. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 1–17, 2024.

[25] B. Mlodozeniec, I. Reid, S. Power, D. Krueger, M. Erdogdu, R. E. Turner, and R. Grosse. Distributional training data attribution. *ArXiv: 2506.12965*, 2025.

[26] S.-Y. Wang, A. Hertzmann, A. A. Efros, R. Zhang, and J.-Y. Zhu. Fast data attribution for text-to-image models. *ArXiv: 2511.10721*, 2025.

[27] A. Alberti, K. Hasanaliyev, M. Shah, and S. Ermon. Data unlearning in diffusion models. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2025.

[28] A. Heng and H. Soh. Selective amnesia: a continual learning approach to forgetting in deep generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 17170–17194. 2023.

[29] A. Golatkar, A. Achille, and S. Soatto. Eternal sunshine of the spotless net: selective forgetting in deep networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9301–9309, 2020.

[30] R. Gandikota, J. Materzyńska, J. Fiotto-Kaufman, and D. Bau. Erasing concepts from diffusion models. In *Proc. of the IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, pages 2426–2436, 2023.

[31] J. Wu, T. Le, M. Hayat, and M. Harandi. Erasing undesirable influence in diffusion models. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 28263–28273, 2025.

[32] Y. Zhang, X. Chen, J. Jia, Y. Zhang, C. Fan, J. Liu, M. Hong, K. Ding, and S. Liu. Defensive unlearning with adversarial training for robust concept erasure in diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 36748–36776. 2024.

[33] Q. Shi, C. Jin, J. Zhang, and Y. Gu. ReTrack: data unlearning in diffusion models through redirecting the denoising trajectory. *ArXiv: 2509.13007*, 2025.

[34] J. Ho and T. Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications (DGMs and Applications)*, 2021.

[35] B. Liu and P. Stone. Continual learning and private unlearning. In *Proc. of the Conf. on Lifelong Learning Agents (CoLLAs)*, pages 243–254, 2022.

[36] R. Zhang, L. Lin, Y. Bai, and S. Mei. Negative preference optimization: from catastrophic collapse to effective unlearning. In *Proc. of the Conf. on Language Modeling (COLM)*, 2024.

[37] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):13046–13055, 2024.

[38] M. Kurmanji, P. Triantafillou, J. Hayes, and E. Triantafillou. Towards unbounded machine unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1957–1987. 2023.

[39] A. K. Veldanda, S.-X. Zhang, A. Das, S. Chakraborty, S. Rawls, S. Sahu, and M. Naphade. LLM surgery: efficient knowledge unlearning and editing in large language models. *ArXiv: 2409.13054*, 2024.

[40] J. Ren, Z. Dai, X. Tang, H. Liu, J. Zeng, Z. Li, R. Goutam, S. Wang, Y. Xing, Q. He, and H. Liu. A general framework to enhance fine-tuning-based LLM unlearning. *Findings of the Association for Computational Linguistics (ACL)*, pages 18464–18476, 2025.

[41] A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[42] P. Liao, X. Li, X. Liu, and K. Keutzer. The ArtBench dataset: benchmarking generative models with artworks. *ArXiv: 2206.11404*, 2022.

[43] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig. From captions to visual concepts and back. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1473–1482, 2015.

[44] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 8748–8763, 2021.

[45] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, pages 4195–4205, 2023.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 32. 2019.

[47] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2019.

[48] M Oquab, T. Darcet, T. Moutakanni, H. Y. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2023.

[49] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[50] E. Pizzi, S. D. Roy, S. N. Ravindra, P. Goyal, and M. Douze. A self-supervised descriptor for image copy detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14532–14542, 2022.

[51] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[52] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[53] S. J. Mason and N. E. Graham. Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: statistical significance and interpretation. *Quarterly Journal of the Royal Meteorological Society*, 128(584):2145–2166, 2002.

[54] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531, 2010.

[55] J. Pearl. *Causality*. Cambridge University Press, 2nd edition, 2013.

[56] J. R. Epifano, R. P. Ramachandran, A. J. Masino, and G. Rasool. Revisiting the fragility of influence functions. *Neural Networks*, 162:581–588, 2023.

[57] Y. Hu, P. Hu, H. Zhao, and J. W. Ma. Most influential subset selection: challenges, promises, and beyond. In *Advances in Neural Information Processing Systems (NeurIPS)*, 37, pages 119778–119810. 2024.

[58] K. K and A. Søgaard. Revisiting methods for finding influential examples. *ArXiv: 2111.04683*, 2021.

[59] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski. Vision transformers need registers. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2023.

A Supplementary Methodology

A.1 Pseudo-Code

Below we provide a Python-style pseudo-code of MUCS. Variables follow the notation in the main text (Sec. 2). The full code is released at https://github.com/sony/mucs.

```
def MUCS(x_hat, c_hat, M_1, L, Z, lambda=0.2):
    # --- Constants ---
    b = 100                                          # Batch size
    num_null = 20                                    # To estimate L_null
    epsilon = 0.001                                  # Prevent division by 0
    # --- Estimate L_null ---
    M_0 = get_copy(M_1)
    M_0 = random_init(M_0)                           # Randomly initialize
    L_null = 0
    for _ in range(num_null):
        x, c = sample_from(Z, b)                     # Get x,c in batches (b)
        # Get model predictions
        sigma, n = get_noise(b)                      # Original noise setup
        x_pred = M_0(x, c, sigma, n)
        # Get losses
        L_est = L(x_pred, x)                         # Original loss setup
        # Accumulate
        L_null += sum(L_est)                         # Sum over batch dim
    L_null /= b * num_null
    # --- Unlearn x_hat ---
    M_2 = get_copy(M_1)
    x_hat, c_hat = repeat(x_hat, c_hat, b)           # Repeat for batch size
    while True:
        x, c = sample_from(Z, b)                     # Get x,c in batches (b)
        # Get model predictions
        sigma, n = get_noise(b)                      # Original noise setup
        x_pred = M_2(x, c, sigma, n)
        sigma, n = get_noise(b)
        x_hat_pred = M_2(x_hat, c_hat, sigma, n)
        # Get losses
        L_ft = L(x_pred, x)                          # Original loss setup
        L_ft = mean(L_ft)                            # Average over batch dim
        L_ga = min(L(x_hat_pred, x_hat), L_null)
        L_ga = mean(L_ga)
        L_mucs = L_ft - lambda * L_ga
        # Optimize
        M_2 = optimization_step(M_2, L_mucs)         # Original optimizer
        # Finished?
        if L_ga >= 0.95 * L_null:
            break
    # --- Compute attribution scores ---
    _, n = get_noise(b)                              # Original noise setup
    sigma, _ = get_noise_gen_sched(b)                # Generation schedule
    A_mucs = empty_array(len(Z))
    for i in range(len(Z)):
        x, c = repeat(Z[i], b)                       # Repeat for num. noises
        # Get model predictions
        x_2_pred = M_2(x, c, sigma, n)
        x_1_pred = M_1(x, c, sigma, n)
        # Get losses
        L_2 = L(x_2_pred, x)                         # Original loss setup
        L_1 = L(x_1_pred, x)
        # Obtain normalized skew
        A_mucs[i] = (L_2 - L_1) / (abs(L_2) + abs(L_1) + epsilon)
        A_mucs[i] = mean(A_mucs[i])                  # Average over noises
    # --- Done! ---
    return A_mucs
```

A.2 Measuring Attribution Performance

The evaluation of TDA algorithms for diffusion models is a challenging task [2]. Some popular options are perceptual assessments, the linear datamodeling score, or indirect evaluation on related but different tasks. Perceptual assessments do not scale and, in general, perception might not correlate with model's behavior [10, 55]. The linear datamodeling score presents a large number of issues discouraging its use [56, 57]. Indirect evaluation is not addressing attribution itself, and is often performed on synthetic/toy problems which may not represent a real-world situation [2, 58]. In this paper, we evaluate TDA performance using a leave-k-out counterfactual setup [2, 8], in which one tests the ability of a model to generate a previously generated image after retraining from scratch without including its influential items in the training set. Albeit time-consuming, we believe it is the only reliable way to assess the performance of TDA algorithms in diffusion models (we introduce a slight modification to parallelize the computational cost of this evaluation, see below).

Given a training dataset $\mathcal{Z}$ and a diffusion model F_1 trained with it, we generate $\hat{\mathbf{z}}$ using some random seed u and a conditioning $\mathbf{c}$ from the test set. Next, an attribution algorithm A provides attribution scores a_i for each $\mathbf{z}_i \in \mathcal{Z}$. These scores are then ranked from highest to lowest and the instances with top-k scores are removed from $\mathcal{Z}$, creating $\mathcal{Z}'$. We retrain from scratch the same model F on $\mathcal{Z}'$, producing F_1', and generate $\hat{\mathbf{z}}'$ using the same u and $\mathbf{c}$ used for $\hat{\mathbf{z}}$. Because of the seed-consistency property of diffusion models [3, 8], $\hat{\mathbf{z}}$ and $\hat{\mathbf{z}}'$ will show a high similarity when A fails to identify/attribute influential items (e.g., when A just randomly selects training items; examples are available in [3, 8] and in Fig. 1). Only when relevant influential samples have been removed from $\mathcal{Z}_{\text{train}}$, the model F_1' will generate dissimilar[4] instances $\hat{\mathbf{z}}'$ under the same u and $\mathbf{c}$ (examples are available in [3, 8] and in Figs. 3–5). To avoid retraining a model for each $\hat{\mathbf{z}}$, we can speed up the previous procedure by conducting m generations in parallel and removing mk items from $\mathcal{Z}$.

By comparing $\hat{\mathbf{z}}$ and $\hat{\mathbf{z}}'$ generations, we can have a nice picture of the performance of an attribution algorithm: a low similarity highlights the degree to which the removed items impede the generation of $\hat{\mathbf{z}}$. To objectively measure image similarity, and to avoid any bias towards a particular metric, we rely on four different but complementary measures: the structural similarity index measure (SSIM) [49], cosine similarity in the SSCD copy-detection space (SSCD) [50], the learned perceptual image patch similarity (LPIPS) measure [51], and cosine similarity in the semantic CLIP space (CLIP) [44]. Fig. 2 shows examples of the distance distributions we get for both random attribution and a considered approach. In the figure, we see that SSIM values concentrate between 0.8 and 1 for the case of random removal, and that MUCS can shift the distance distribution to values between 0.4 and 0.9.

Our final assessment is based on comparing the similarity distribution of the considered algorithms A with the one obtained by random removal A_{rand} (our reference similarity distribution). In particular, we are interested in a low overlap of the two distributions, indicating that removing training instances according to A had a meaningful effect. To objectively quantify the overlap between two distributions, we use the area under the ROC curve (AUC) measure, which is a common evaluation measure employed for that purpose [52], and which has the nice property of being proportional to the U statistic of the Mann-Whitney U test for measuring whether two samples correspond to the same distribution [53]. This way, a good attribution algorithm should achieve an AUC close to 1, corresponding to a low overlap, indicating that attributed items' removal is affecting generation.

Our final setup involves removing $k = 2\%$ of the training data for $m = 20$ generations, and running it 6 times using different network and seed initializations. Hence, AUC calculations are made on pairs of 120 generated items (random and considered approach). Notice that removing km items corresponds to removing a maximum of 40% of the training data (there can be overlaps between the m items). Notice also that, ideally, the value of k should depend on each dataset and generated item. Here, it was empirically set such that AUCs were significantly larger than random, and such that less than half of the training data was removed.

A.3 Model, Training, and Generation

Architecture — As mentioned in the main text, we use a DiT/B architecture [45], which consists of 12 Transformer blocks of 768 channels, 12 heads, an MLP ratio of 4, and a dropout of 0.1. To

[4] Intuitively, removing influential items here corresponds to removing nearest neighbors in the distribution 'seen' by the model, hence unpopulating the region where the generated sample lies. When that happens, the newly trained model is nudged to generate a different item given the same seed (and conditioning).

adapt different image resolutions to this architecture, we use two residual convolutional blocks with a number of channels and strides that depend on the dataset (Table 4). These blocks use a kernel of 5 with same-size padding, instance norm, and the SiLU activation (the decoder blocks use transposed convolutions). After image encoding with this structure, and together with positional encoding, we add 32 extra tokens to be used as memory registers [59].

The conditioning, as well as the σ embeddings, are processed by a two-layer MLP with SiLU activations and a dropout of 0.1. The number of channels of the conditioning module varies depending on the density of the information contained in the conditioning vector (compare ArtBench10 and COCO in Table 4). For conditional models, we train with a condition drop of 10% to enable classifier-free guidance (CFG) at sampling. Note that, as mentioned in the main text, we consider CIFAR10 as unconditional, ArtBench10 is conditioned on the class identifier, and COCO is conditioned on CLIP-Text embeddings. All network weights are initialized following PyTorch's defaults [46].

Diffusion Variant and Generation — We consider the EDM diffusion variant [16], using $P_{\text{mean}} = -1.2$, $P_{\text{std}} = 1.2$, and $\sigma_{\text{data}} = 0.5$ for all datasets (data is normalized between -1 and 1). As illustrated in the main text, the loss includes the characteristic EDM weighting term (Sec. 2.1). For generation, we use $\sigma_{\text{min}} = 2 \cdot 10^{-3}$, $\sigma_{\text{max}} = 80$, $\rho = 7$, and 32 steps of the 2nd order Heun sampler (we use deterministic sampling). The CFG weight depends on the dataset (see Table 4).

Training — The training of our diffusion models consists of hundreds of thousands of weight updates using AdamW [47] with a variable learning rate and a weight decay of 0.01 (details about number of updates and nominal learning rate for each dataset can be found in Table 4). We perform a linear learning rate warmup of 10 k steps and maintain it constant for the rest of the updates. Our batch size is 128, and we use an exponential moving average of the model's weights [16] with a momentum of 0.999.

Table 4: Configuration differences for each dataset. The rest of the parameters follow the canonical DiT/B and EDM setups (see text).

Parameter	**Dataset**		
	CIFAR10	ArtBench10	COCO
Image resolution	32×32	64×64	64×64
Encoder/decoder channels	[4,4]	[16,16]	[16,16]
Encoder/decoder strides	[1,1]	[1,2]	[1,2]
Conditioning	None	Class ID	CLIP-Text
Conditioning block channels	256	256	768
Total learnable parameters	100 M	100 M	146 M
Learning rate	10^{-4}	$2{\cdot}10^{-4}$	$2{\cdot}10^{-4}$
Pre-training updates	200 k	250 k	300 k
CFG weight	N/A	2.0	3.0

A.4 Rank-based TDA Ensemble

To produce a TDA ensemble, we take the scores **a** of every approach and transform them into ranks **r** following a descending order. We then weight **r** and sum across the approaches that form the ensemble. Weights are loosely assigned based on the average performances reported in Table 1 in the main paper. Following that, we set the weight of MUCS to 10, DAS to 7, D-TRAK to 6, DINO to 5, CLIP to 5, Condition to 3, Forward-INF to 2, and AbU to 0. After summing the weighted rankings, we take the inverse as a new attribution score (descending order) and evaluate. In preliminary experiments, we employed a simple 0–1 normalization instead of ranks, but we obtained slightly worse results.

We acknowledge that such an ensembling procedure can be further improved, but leave this task for future works. In this paper, we just aim to perform an initial exploration of the suitability of ensembling different attribution methods, and we believe the considered approach is sufficient for that purpose. To the best of our knowledge, this is the first time that such an ensembling approach

has been considered for TDA. As mentioned in the main paper, our contribution is to show that it can provide important boosts in some cases, but that one should be careful in others (Sec. 5).

B Additional Results

B.1 Comparison with Existing Approaches

To complement the results in the main paper, we here show the average normalized similarity difference between A_{rand} and each considered approach (Table 5). We calculate the average normalized similarity difference as

$$\frac{100}{mk} \sum_{i=1,\dots mk} \left(\frac{s_i}{\text{median}(\mathcal{S})} - 1 \right),$$

where $s_i = S(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i)$ denotes the corresponding image similarity metric and $\mathcal{S}$ denotes the set of all A_{rand} vs. A similarities (see Sec. 4 and Appendix A.2). In the case of LPIPS, we convert its loss L_{LPIPS} into a similarity measure s by simply using $s = \max(-1, 1 - L_{\text{LPIPS}})$.

Apart from assessing distribution overlaps, we also report the amortized running time of the considered TDA approaches (Table 6). Finally, our results also include a sample of the images generated for each dataset, before and after attribution-based removal, comparing reference (generated with F_1) and regenerated (generated with F_2) images (Figs. 3–5).

Table 5: Comparison with existing approaches on CIFAR10 (top), ArtBench10 (middle), and COCO (bottom). Approaches can be categorized into model-agnostic (MA), influence function-based (IF), and unlearning-based (U). We report average similarity differences (%, the lower the better) for the considered measures, plus their average (μ; last column). The $\pm$ signs indicate 95% confidence intervals.

	Approach	Categ.	SSIM	SSCD	LPIPS	CLIP	μ
CIFAR10	Condition	MA	−2.5±0.9	−4.5±1.7	−2.6±1.4	−3.6±1.9	−3.3±1.5
	CLIP	MA	−4.8±1.4	−8.9±2.2	−6.1±1.7	−7.8±2.3	−6.9±1.9
	DINO	MA	−4.5±1.4	−7.5±2.1	−5.5±1.7	−6.4±2.3	−6.0±1.9
	D-TRAK [4]	IF	−8.1±1.9	−11.0±2.4	−9.7±2.1	−7.3±2.3	−9.0±2.2
	DAS [5]	IF	−11.2±2.1	−15.5±2.6	−13.1±2.2	−11.7±2.4	−12.9±2.3
	Forward-INF [7]	U	−3.6±1.5	−3.6±2.0	−2.9±1.7	−2.5±1.9	−3.1±1.8
	AbU [8]	U	−2.8±1.2	−2.1±1.7	−1.3±1.4	−0.8±1.7	−1.7±1.5
	MUCS (proposed)	U	−13.5±2.1	−20.8±2.3	−17.9±1.8	−18.1±2.1	−17.6±2.1
ArtBench10	Condition	MA	−3.0±1.3	−6.7±2.9	−4.0±1.8	−2.9±1.3	−4.2±1.8
	CLIP	MA	−5.2±1.6	−12.5±3.3	−7.6±2.0	−7.3±1.8	−8.1±2.2
	DINO	MA	−3.6±1.5	−8.5±3.1	−4.6±1.8	−4.1±1.5	−5.2±2.0
	D-TRAK [4]	IF	−10.5±2.0	−20.9±3.9	−13.4±2.4	−9.4±2.1	−13.6±2.6
	DAS [5]	IF	−11.1±2.0	−21.3±3.6	−14.1±2.3	−9.0±2.0	−13.9±2.5
	Forward-INF [7]	U	−6.4±1.7	−11.7±3.3	−7.6±2.0	−5.4±1.8	−7.8±2.2
	AbU [8]	U	−3.2±1.3	−5.0±2.8	−3.3±1.7	−2.9±1.4	−3.6±1.8
	MUCS (proposed)	U	−12.8±1.9	−25.4±3.2	−17.0±2.1	−11.9±1.7	−16.8±2.2
COCO	Condition	MA	−7.4±2.3	−12.8±4.3	−5.5±2.2	−5.4±1.8	−7.8±2.7
	CLIP	MA	−11.8±3.0	−21.4±5.1	−11.7±2.9	−12.1±2.3	−14.3±3.3
	DINO	MA	−11.9±2.9	−21.5±4.9	−11.0±2.8	−10.8±2.0	−13.8±3.2
	D-TRAK [4]	IF	−13.3±2.8	−18.1±4.3	−12.4±2.7	−6.8±1.7	−12.6±2.9
	DAS [5]	IF	−13.9±3.0	−20.3±4.7	−12.6±2.9	−7.9±2.0	−13.7±3.2
	Forward-INF [7]	U	−8.2±2.6	−11.6±4.4	−6.4±2.6	−5.6±2.0	−8.0±2.9
	AbU [8]	U	−5.4±2.4	−3.9±3.8	−3.0±2.1	−2.8±1.6	−3.8±2.5
	MUCS (proposed)	U	−15.0±2.9	−24.9±4.8	−16.2±3.0	−8.9±2.0	−16.2±3.2

Table 6: Amortized time (in minutes) to compute attribution for a single generated item, using a single NVIDIA H100 GPU. Time depends on the size of the training dataset and, in this table, includes the proportion of the time to initialize all operations (less than 0.1 minutes).

Approach	**CIFAR10**	**ArtBench10**	**COCO**
Random	<0.1	<0.1	0.1
Condition	<0.1	<0.1	0.2
CLIP	0.1	0.1	0.3
DINO	<0.1	<0.1	0.2
D-TRAK	57.4	36.0	101.1
DAS	66.7	40.8	113.9
Forward-INF	34.3	36.1	82.8
AbU	36.4	38.1	87.9
MUCS	22.8	24.2	56.5

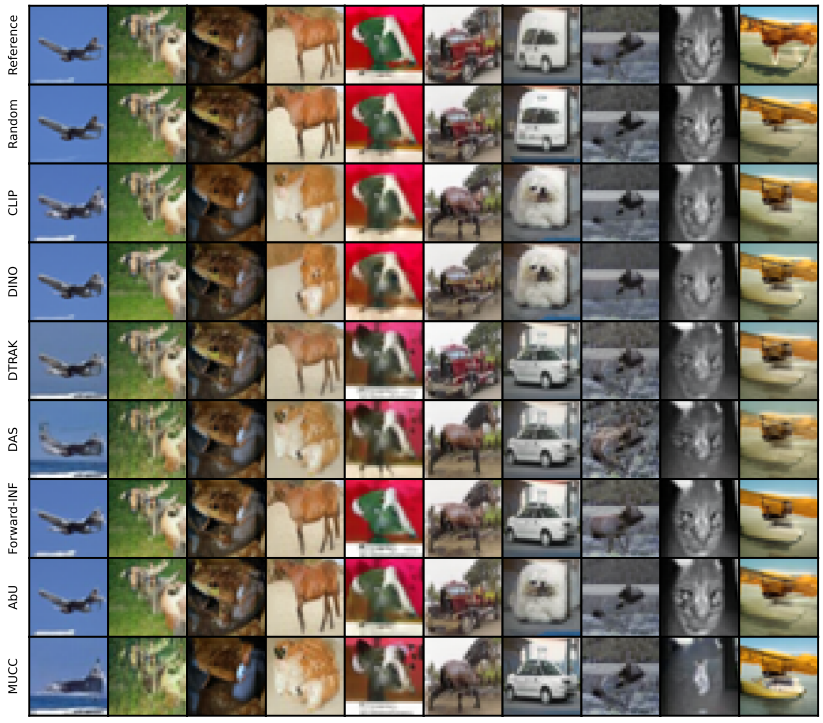

Figure 3: Reference (first row) and regenerated (other rows) images for the considered approaches on CIFAR10.

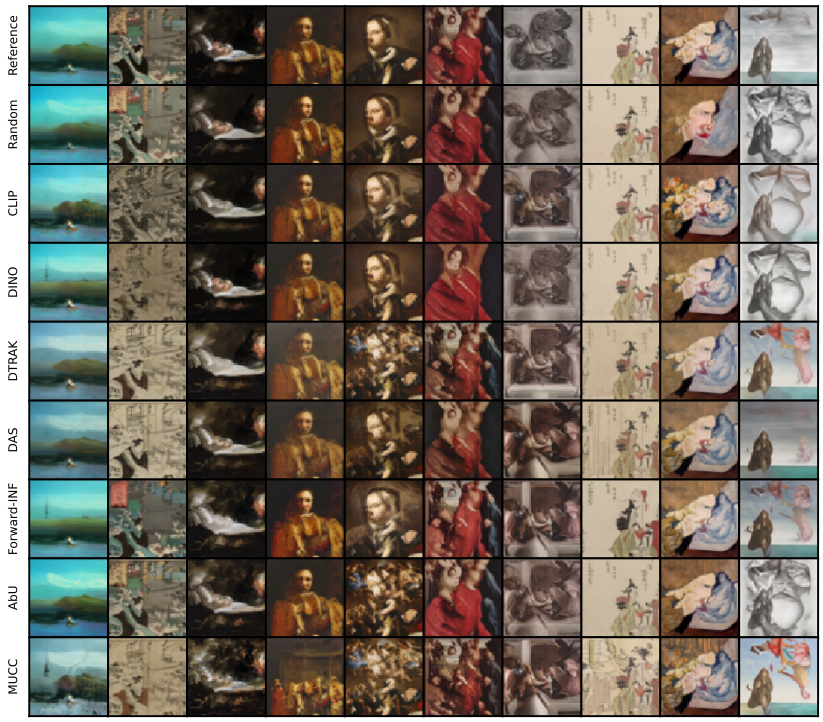

Figure 4: Reference (first row) and regenerated (other rows) images for the considered approaches on ArtBench10.



Figure 5: Reference (first row) and regenerated (other rows) images for the considered approaches on COCO.

B.2 Overlap Between Generated Items

Table 7 shows the average overlap between top-k attributed instances. First, for m generated items, we obtain one top-k array of topmost influential training instances. Next, independently for every attribution approach, we compute the overlap (intersection) across items, which results in $m(m-1)/2$ combinations. Finally, we take the average across those combinations.

We see that there is always an overlap (that is, overlap never significantly close to 0), and that this usually ranges between 2 and 7% of the top-k instances. The relatively low values for the latter indicate that the removed items are generally different for the m generated instances. It also indicates that the attribution methods do not exploit any kind of massive generative model disruption (e.g., targeting only and always the same 'hub'/structural training items for every generated instance).

Table 7: Average top-k attributed instances' overlap (%) between generated items. The $\pm$ signs denote 95% confidence intervals.

Approach	**CIFAR10**	**ArtBench10**	**COCO**
Random	2.0 ± 0.0	2.0 ± 0.0	2.0 ± 0.0
Condition	2.0 ± 0.1	1.9 ± 0.1	2.5 ± 0.2
CLIP	5.9 ± 0.2	6.5 ± 0.2	4.1 ± 0.2
DINO	6.0 ± 0.2	6.1 ± 0.2	3.4 ± 0.2
D-TRAK	3.6 ± 0.2	2.6 ± 0.2	3.8 ± 0.2
DAS	3.9 ± 0.3	2.1 ± 0.2	2.3 ± 0.1
Forward-INF	10.9 ± 0.4	2.9 ± 0.2	3.0 ± 0.2
AbU	32.0 ± 0.8	39.6 ± 0.8	25.9 ± 0.6
MUCS	2.7 ± 0.2	3.0 ± 0.2	4.2 ± 0.2

B.3 Ensembling Approaches

Continuing with the overlap idea, we can compute top-k instance overlap between TDA approaches. Comparing pairwise, this tells us how different are the top-k selected items across approaches (Table 8). Interestingly, we see that the overlap is usually below 35%, which suggests that an ensemble approach could benefit from these multiple criteria and obtain better performance. After running the aforementioned rank-based ensemble approach (Appendix A.4), we obtain mixed results depending on the dataset (Table 9; see the comments in the main paper).

Table 8: Average top-k overlap (%) between methods for ArtBench10, comparing each generated item and averaging (see text).

Approach $\downarrow \rightarrow$	Rand	Cond	CLIP	DINO	DTRK	DAS	F-INF	AbU	MUCS
Random	-	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
Condition	2.0	-	8.1	6.3	13.4	16.2	17.4	4.2	10.3
CLIP	2.0	8.1	-	28.2	13.6	13.9	6.8	2.9	19.9
DINO	2.0	6.3	28.2	-	9.8	9.9	5.6	2.4	14.6
D-TRAK	2.0	13.4	13.6	9.8	-	60.7	12.2	6.9	36.0
DAS	2.0	16.2	13.9	9.9	60.7	-	13.3	5.6	32.5
Forward-INF	2.0	17.4	6.8	5.6	12.2	13.3	-	8.0	8.5
AbU	2.0	4.2	2.9	2.4	6.9	5.6	8.0	-	6.3
MUCS	2.0	10.3	19.9	14.6	36.0	32.5	8.5	6.3	-

C Further Information

C.1 Licenses

As mentioned in the main text, we use the CIFAR10 [41], ArtBench10 [42], and COCO [43] datasets. CIFAR10 is under MIT license, ArtBench10 is under Fair Use license, COCO images are under

Table 9: MUCS (R) and ensemble results (A–D) for CIFAR10 (top), ArtBench10 (middle), and COCO (bottom).

#	Weight in the Ensemble							AUC			
	MUCS	DAS	D-TRAK	DINO	CLIP	Cond	F-INF	**SSIM**	**SSCD**	**LPIPS**	**CLIP**
R	10	0	0	0	0	0	0	0.889	0.901	0.930	0.868
A	10	7	6	5	5	3	2	0.825	0.843	0.864	0.803
B	10	0	0	5	5	3	0	0.696	0.706	0.734	0.716
C	0	7	6	5	5	3	2	0.775	0.788	0.805	0.761
D	0	0	0	5	5	3	0	0.593	0.612	0.613	0.604
R	10	0	0	0	0	0	0	0.850	0.840	0.859	0.823
A	10	7	6	5	5	3	2	0.858	0.858	0.878	0.843
B	10	0	0	5	5	3	0	0.780	0.816	0.821	0.835
C	0	7	6	5	5	3	2	0.797	0.803	0.824	0.815
D	0	0	0	5	5	3	0	0.682	0.712	0.706	0.736
R	10	0	0	0	0	0	0	0.750	0.763	0.799	0.715
A	10	7	6	5	5	3	2	0.784	0.803	0.822	0.795
B	10	0	0	5	5	3	0	0.776	0.803	0.829	0.795
C	0	7	6	5	5	3	2	0.743	0.763	0.782	0.758
D	0	0	0	5	5	3	0	0.715	0.743	0.757	0.806

Flickr Terms of Use license (various Creative Commons), and COCO captions is under the Creative Commons Attribution 4.0 license. As for pre-trained models, SSIM [49], SSCD [50], and CLIP [44] are under MIT license, LPIPS [51] is under BSD 2-Clause license, and DINOv2 [48] is under Apache-2.0 license. We release our code under Apache-2.0 license.

C.2 Compute Resources

Training one of the considered diffusion models takes between 1 and 2 days, using a single NVIDIA H100 GPU with 80 GB memory (depending if the dataset is small, like CIFAR10, or larger, like COCO). Computing attribution for 20 items takes a maximum of 1 day using the same resource (one full forward pass per training instance, repeated 20 times; see also Table 6). The time to evaluate results is just seconds. Hence, the full pipeline (training F_1, computing A, training F_2, and evaluating) takes roughly 3 to 5 days.

C.3 Limitations and Broader Impact

While MUCS demonstrates strong empirical performance, its theoretical guarantees remain limited, as is common in the TDA literature. Attribution reliability may still degrade with different, larger-scale datasets, or with highly redundant or overly sparse training data. Our evaluation is also restricted to diffusion models and image data, and generalization to other generative paradigms or domains is not guaranteed. Additionally, the absence of ground-truth attribution scores makes absolute performance assessment inherently approximate, and the computational overhead of performing a full forward pass for each training instance may limit applicability at very large scales or in real-world situations.

TDA has the potential to significantly improve the transparency and accountability of generative models. From a socio-technical perspective, reliable TDA could empower anyone with access to training data and model weights to audit the influence of specific data points on generated outputs, with direct implications for intellectual property, consent, and data governance frameworks. Improved unlearning capabilities could further support right-to-be-forgotten compliance under regulations such as GDPR. However, TDA tools could also be misused to reverse-engineer proprietary datasets or circumvent data protection measures, raising dual-use concerns. As generative AI becomes more pervasive, tools like MUCS could help mitigate harms from non-consensual data use, but their deployment should be accompanied by appropriate governance structures to avoid over-reliance on purely technical solutions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and introduction contain a concise summary of the paper's contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Sec. 5 and Appendix C.3.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Both the main document and the Appendix disclose all the information.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All the data is publicly available, pseudo-code is provided in Appendix A.1, and the full code will be released under Apache-2.0 license after the review phase.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Sec. 4 and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We use both statistical significance (main paper) and confidence intervals (Appendix).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and checked the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix C.3. Also partially addressed along the main paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Appendix C.1.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: There is pseudocode in Appendix A.1 and the released code will provide a comprehensive README file.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.