Data Attribution for Text-to-Image Models by Unlearning Synthesized Images

Sheng-Yu Wang[1]   **Aaron Hertzmann**[2]   **Alexei A. Efros**[3]   **Jun-Yan Zhu**[1]   **Richard Zhang**[2]

[1]Carnegie Mellon University   [2]Adobe Research   [3]UC Berkeley

Abstract

The goal of data attribution for text-to-image models is to identify the training images that most influence the generation of a new image. We can define "influence" by saying that, for a given output, if a model is retrained from scratch without that output's most influential images, the model should then fail to generate that output image. Unfortunately, directly searching for these influential images is computationally infeasible, since it would require repeatedly retraining from scratch. We propose a new approach that efficiently identifies highly-influential images. Specifically, we simulate *unlearning the synthesized image*, proposing a method to increase the training loss on the output image, without catastrophic forgetting of other, unrelated concepts. Then, we find training images that are forgotten by proxy, identifying ones with significant loss deviations after the unlearning process, and label these as influential. We evaluate our method with a computationally intensive but "gold-standard" retraining from scratch and demonstrate our method's advantages over previous methods.

1 Introduction

Data attribution for text-to-image generation aims to identify which training images "influenced" a given output. The black-box nature of state-of-the-art image generation models [1, 2, 3, 4, 5, 6, 7, 8], together with the enormous datasets required [9], makes it extremely challenging to understand the contributions of individual training images. Although generative models can, at times, replicate training data [10, 11], they typically create samples distinct from any specific training image.

We believe that a counterfactual definition of "influence" best matches the intuitive goal of attribution [12, 13]. Specifically, we say that a collection of training images is influential for a given output image if: removing those images from the training set and then retraining from scratch makes the model unable to generate the synthesized image. Unfortunately, directly searching for the most influential images according to this definition is computationally infeasible since it would require training an exponentially large number of new models from scratch.

Hence, practical influence estimation requires effective approximations. For example, many approaches replace retraining with a closed-form approximation, computed separately for each training image [12, 14, 15, 13, 16]. For text-to-image attribution, these methods are outperformed by simple matching of off-the-shelf image features [17]. Wang *et al.* [18] use customization to study the effect of training a model towards an exemplar, but find limited generalization to the general large-scale training case. We aim for a tractable method that accurately predicts influence according to the counterfactual definition.

We propose an approach to influence prediction with two key ideas (Figure 1). First, we can approximate removing a training image from a model by an optimization that we call *unlearning*. Unlearning increases the training loss of the target image while protecting unrelated concepts; we can then compute training loss for the original synthesized image. However, directly applying this idea
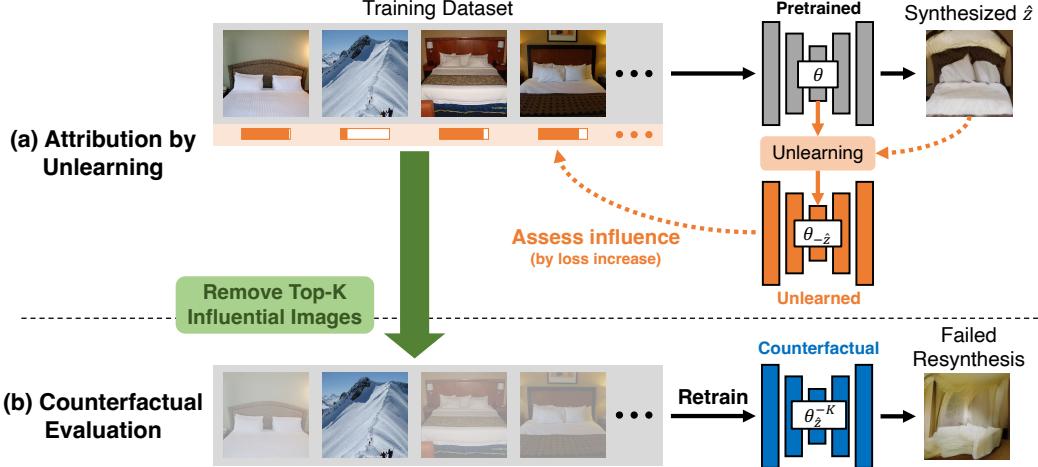
Figure 1: (a) **Our algorithm**: We propose a new data attribution method using machine unlearning. By modifying the pretrained model θ to unlearn the synthesized result $\hat{\mathbf{z}}$, the model also forgets the influential training images crucial for generating that specific result. (b) **Evaluation**: We validate our method through counterfactual evaluation, where we retrain the model without the top K influential images identified by our method. When these influential images are removed from the dataset, the model fails to generate the synthesized image.

would require unlearning separately for each training image. Our second main idea is to reverse the roles: we *unlearn the synthesized image, and then evaluate which training images are represented worse by the new model.* This requires only one unlearning optimization, rather than a separate unlearning for each training image.

The methodology for unlearning is important. Unlearning a synthesized image by naively maximizing its loss leads to catastrophic forgetting [19], where the model fails to generate other unrelated concepts as well. Inspired by work on unlearning data for classifiers [20, 21], we mitigate this issue by regularizing gradient directions using Fisher information to retain pretrained information. Additionally, we find that updating only the key and value mappings in the cross-attention layers improves attribution performance. We show how "influence functions" [13, 15] can be understood as approximations to unlearning in Appendix A, but they are limited by their closed-form nature.

We perform a rigorous counterfactual validation: removing a predicted set of influential images from the training set, retraining from scratch, and then checking that the synthesized image is no longer represented. We use MSCOCO [22] (∼100k images), which allows for retraining models within a reasonable compute budget. We also test on a publicly-available attribution benchmark [18] using customized text-to-image models [23]. Our experiments show that our algorithm outperforms prior work on both benchmarks, demonstrating that unlearning synthesized images is an effective way to attribute training images. Our code is available at: `https://peterwang512.github.io/AttributeByUnlearning`.

In summary, our contributions are:

- We propose a novel method for data attribution for text-to-image models, unlearning the synthesized image and identifying which training images are forgotten.

- We find and ablate the components for making unlearning efficient and effective, employing Fisher information and tuning a critical set of weights.

- We rigorously show that our method is counterfactual predictive by omitting influential images, retraining, and checking that the synthesized image cannot be regenerated. Along with the existing Customized Model benchmark, we show our method identifies influential images more effectively than recent baselines based on customization and influence functions.

2 Related Work

Attribution. *Influence functions* [15, 13] approximate how the objective function of a test datapoint would change after perturbing a training datapoint. One may then predict attribution according to the training points that can produce the largest changes. Koh and Liang [13] proposed using influence functions to understand model behavior in deep discriminative models. The influence function requires calculating a Hessian of the model parameters, for which various efficient algorithms have been proposed, such as inverse hessian-vector products [13], Arnoldi iteration [24], Kronecker factorization [25, 26, 27], Gauss-Newton approximation [16], and nearest neighbor search [28].

Other methods explore different approaches. Inspired by the game-theory concept of Shapley value [29], several methods train models on subsets of training data and estimate the influence of a training point by comparing the models that had that training point in their data to those that did not [30, 31, 32, 33]. Pruthi et al. [34] estimate influence by tracking train-test image gradient similarity over the course of model training.

Recent methods perform attribution for diffusion-based image generation. Wang *et al.* [18] proposed attribution by model customization [35, 23], where a pretrained model is influenced by tuning towards an exemplar concept. Several works adapt TRAK [16], an influence function-based method, to diffusion models, extending it by attributing at specific denoising timesteps [12], or by improving gradient estimation and using Tikhonov regularization [14]. Unlike these methods, our method performs attribution by directly unlearning a synthesized image and tracking the effect on each training image. Our method outperforms existing methods in attributing both customized models and text-to-image models.

Machine unlearning. Machine unlearning seeks to efficiently "remove" specific training data points from a model. Recent studies have explored concept erasure for text-to-image diffusion models, specified by a text request [36, 37, 38, 39, 40], whereas we remove individual images. While forgetting may be achieved using multiple models trained with subsets of the dataset beforehand [41, 42, 43], doing so is prohibitively expensive for large-scale generative models.

Instead, our approach follows unlearning methods that update model weights directly [20, 44, 45, 46, 21]. The majority of prior methods use the Fisher information matrix (FIM) to approximate retraining without forgetting other training points [20, 45, 47, 21, 48, 49]. In particular, we are inspired by the works from Guo *et al.* [20] and Tanno *et al.* [21], which draw a connection between FIM-based machine unlearning methods and influence functions. We show that unlearning can be efficiently applied to the attribution problem, by "unlearning" output images instead of training data.

Replication detection. Shen *et al.* [50] identify repeated pictorial elements in art history. Somepalli *et al.* [11] and Carlini *et al.* [10] investigate text-to-image synthesis of perceptually-exact copies of training images. Unlike these works, our work focuses on data attribution for more general synthesis settings beyond replication.

3 Problem Setting and Evaluation

Our goal is to attribute a generated image to its training data. We represent the training data as $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{c}_i)\}_{i=1}^{N}$, where $\mathbf{x} \in \mathcal{X}$ denotes images and $\mathbf{c}$ represents the conditioning text. A learning algorithm $\mathcal{A} : \mathcal{D} \rightarrow \theta$ yields parameters of a generative model; for instance, $\theta = \mathcal{A}(\mathcal{D})$ is a model trained on $\mathcal{D}$. We focus on diffusion models that generate an image from a noise map $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. A generated image from text $\mathbf{c}$ is represented as $\hat{\mathbf{x}} = G_\theta(\epsilon, \mathbf{c})$. To simplify notation, we write a text-image tuple as a single entity. A synthesized pair is denoted as $\hat{\mathbf{z}} = (\hat{\mathbf{x}}, \mathbf{c})$, and a training pair is denoted as $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{c}_i) \sim \mathcal{D}$. We denote the loss of an image $\mathbf{x}$ conditioned on $\mathbf{c}$ as $\mathcal{L}(\mathbf{z}, \theta)$.

Next, we describe the "gold-standard" evaluation method that we use to define and evaluate influence. Section 4 describes our method for predicting influential images.

Counterfactual evaluation. A reliable data attribution algorithm should accurately reflect a *counterfactual prediction*. That is, if an algorithm can identify a set of truly influential training images, then a model trained *without* those images would be incapable of generating or representing that image. As noted by Ilyas *et al.* [33] and Park *et al.* [16], counterfactual prediction is computationally intensive to validate. As such, these works introduce the Linear data modeling (LDS) score as an

efficient proxy, but with the assumption that data attribution methods are *additive*, which does not hold for feature matching methods and our method.

In our work, we invest substantial computational resources to the "gold standard" counterfactual evaluation within our resource limits. That is, we use an attribution algorithm to identify a critical set of K images, denoted as $\mathcal{D}_{\hat{\mathbf{z}}}^K \subset \mathcal{D}$. We then train a generative model without those images from scratch, *per synthesized sample and per attribution method*. Despite the computational cost, this allows us to provide the community with a direct evaluation of counterfactual prediction, without relying on a layer of approximations. We formalize our evaluation scheme as follows.

Training a counterfactual model. For evaluation, an attribution algorithm is given a budget of K images for attributing a synthesized sample $\hat{\mathbf{z}}$, denoted as $\mathcal{D}_{\hat{\mathbf{z}}}^K$. We then train a leave-$K$-out model $\theta_{\hat{\mathbf{z}}}^{-K}$ from scratch using $\mathcal{D}_{\hat{\mathbf{z}}}^{-K} = \mathcal{D} \backslash \mathcal{D}_{\hat{\mathbf{z}}}^K$, the dataset with the K attributed images removed:

$$\theta_{\hat{\mathbf{z}}}^{-K} = \mathcal{A}(\mathcal{D}_{\hat{\mathbf{z}}}^{-K}), \tag{1}$$

Evaluating the model. We then compare this "leave-K-out" model against $\theta_0 = \mathcal{A}(\mathcal{D})$, the model trained with the entire dataset, and assess how much it loses its capability to represent $\hat{\mathbf{z}}$ in terms of both the loss change $\Delta\mathcal{L}(\hat{\mathbf{z}}, \theta)$ and the capability to generate the same sample $\Delta G_\theta(\epsilon, \mathbf{c})$.

First, if the leave-K-out model is trained without the top influential images, it should reconstruct synthetic image $\hat{\mathbf{z}}$ more poorly, resulting in a higher $\Delta\mathcal{L}(\hat{\mathbf{z}}, \theta)$:

$$\Delta\mathcal{L}(\hat{\mathbf{z}}, \theta) = \mathcal{L}(\hat{\mathbf{z}}, \theta_{\hat{\mathbf{z}}}^{-K}) - \mathcal{L}(\hat{\mathbf{z}}, \theta_0). \tag{2}$$

Second, if properly selected, the leave-K-out model should no longer be able to generate $\hat{\mathbf{x}} = G_\theta(\epsilon, \mathbf{c})$. For diffusion models, in particular, we can rely on the "seed consistency" property [12, 51, 52]. Georgiev *et al.* [12] find that images generated by two independently trained diffusion models from the same random noise ϵ have little variations. They leverage this property to evaluate attribution via $\Delta G_\theta(\epsilon, \mathbf{c})$, the difference of generated images between θ_0 and $\theta_{\hat{\mathbf{z}}}^{-K}$. An effective attribution algorithm should lead to a leave-K-out model generating images that deviate more from the original images, resulting in a larger $\Delta G_\theta(\epsilon, \mathbf{c})$ value:

$$\Delta G_\theta(\epsilon, \mathbf{c}) = d\big(G_{\theta_0}(\epsilon, \mathbf{c}), G_{\theta_{\hat{\mathbf{z}}}^{-K}}(\epsilon, \mathbf{c})\big), \tag{3}$$

where d can be any distance function, such as L2 or CLIP [53]. Georgiev *et al.* [12] also adopt $\Delta G_\theta(\epsilon, \mathbf{c})$ for evaluation. While evaluating loss increases and seed consistency is specific to diffusion models, the overarching idea of retraining and evaluating if a synthesized image is still in the model applies across generative models.

4   Attribution by Unlearning

In this section, we introduce our approach to attribute training images for a text-to-image model $\theta_0 = \mathcal{A}(\mathcal{D})$, trained on dataset $\mathcal{D}$. Our goal is to find the highly influential images on a given synthetic image $\hat{\mathbf{z}}$ in dataset $\mathcal{D}$.

Formally, we define a data attribution algorithm $\boldsymbol{\tau}$, which given access to the training data, model, and learning algorithm, produces a set of scores, denoted by $\boldsymbol{\tau}(\hat{\mathbf{z}}, \mathcal{D}, \theta, \mathcal{A}) \in \mathbb{R}^N$. The highest K influencing images are then selected according to their scores. Due to the training set sizes $\mathcal{D}$ used in text-to-image models, we simplify the problem by individually estimating the influence of each training point: $\boldsymbol{\tau}(\hat{\mathbf{z}}, \mathcal{D}, \theta, \mathcal{A}) = [\tau(\hat{\mathbf{z}}, \mathbf{z}_1), \tau(\hat{\mathbf{z}}, \mathbf{z}_2), \ldots, \tau(\hat{\mathbf{z}}, \mathbf{z}_N)]$, where τ represents the factorized attribution function that estimates influence based on the synthesized sample and a training point. Although τ can access all training data, parameters, and the learning algorithm, we omit them for notational simplicity.

Given infinite compute, one can find critical sets of influential images by training a model from every possible subset of K images from scratch and searching for one that "forgets" the synthesized image the most. However, the search space is combinatoric and infeasible in practice. To address this problem, we have made two modifications. First, rather than training from scratch, we use

4

model unlearning—efficiently tuning a pretrained model to remove a data point. Second, rather than searching through the combinatoric space of training points, we instead apply unlearning to the *synthesized* image and then assess how effectively each training image is also forgotten as a result. As a heuristic to measure the degree of removal, we track the training loss changes for each training image after unlearning, and we find this effective for data attribution.

Unlearning the synthesized image. A naive approach to unlearn a synthesized image $\hat{\mathbf{z}}$ is to solely maximize its loss $\mathcal{L}(\hat{\mathbf{z}}, \theta)$. However, only optimizing for this leads to catastrophic forgetting [19], where the model can no longer represent other concepts.

Instead, we propose to retain the information from the original dataset while "removing" the synthesized image, as though it had been part of training. Given model trained on the original dataset $\theta_0 = \mathcal{A}(\mathcal{D})$ and a synthesized image $\hat{\mathbf{z}}$, we compute a new model $\theta_{-\hat{\mathbf{z}}} = \mathcal{A}(\mathcal{D} \backslash \hat{\mathbf{z}})$, with $\hat{\mathbf{z}}$ removed. Here use the set removal notation $\backslash$ to specify a "negative" datapoint in the dataset. Concretely, we solve for the following objective function, using elastic weight consolidation (EWC) loss [19] as an approximation:

$$
\begin{aligned}
\mathcal{L}_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta) &= -\mathcal{L}(\hat{\mathbf{z}}, \theta) + \sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta) \\
&\approx -\mathcal{L}(\hat{\mathbf{z}}, \theta) + \frac{N}{2}(\theta - \theta_0)^T F(\theta - \theta_0),
\end{aligned}
\tag{4}
$$

where F is the Fisher information matrix, which is approximated as a diagonal form for computational efficiency. F approximates the training data loss to the second-order [54], a technique widely used in continual learning [19, 55]. This enables us to solve for the new model parameters $\theta_{-\hat{\mathbf{z}}}$ efficiently, by initializing from the pretrained model θ_0. We optimize this loss with Newton updates:

$$
\theta \leftarrow \theta + \frac{\alpha}{N} F^{-1} \nabla \mathcal{L}(\hat{\mathbf{z}}, \theta),
\tag{5}
$$

where α controls the step size, F^{-1} is the inverse of the Fisher information matrix. In practice, Newton updates allow us to achieve effective attribution with few iterations and, in some cases, as few as one step. We denote the unlearned model as $\theta_{-\hat{\mathbf{z}}}$. We provide details of the EWC loss and Newton update in Appendix A.

Attribution using the unlearned model. After we obtain the unlearned model $\theta_{-\hat{\mathbf{z}}}$, we define our attribution function τ by tracking the training loss changes for each training sample $\mathbf{z}$:

$$
\tau(\hat{\mathbf{z}}, \mathbf{z}) = \mathcal{L}(\mathbf{z}, \theta_{-\hat{\mathbf{z}}}) - \mathcal{L}(\mathbf{z}, \theta_0).
\tag{6}
$$

The value $\tau(\hat{\mathbf{z}}, \mathbf{z})$ is expected to be close to zero for most unrelated training images since the EWC loss used in obtaining $\theta_{-\hat{\mathbf{z}}}$ acts as a regularization to preserve the original training dataset. A higher value of $\tau(\hat{\mathbf{z}}, \mathbf{z})$ indicates that unlearned model $\theta_{-\hat{\mathbf{z}}}$ no longer represents the training sample $\mathbf{z}$.

Relation with influence functions. Our method draws a parallel with the influence function, which aims to estimate the loss change of $\hat{\mathbf{z}}$ by removing a training point $\mathbf{z}$. However, training leave-one-out models on every training point is generally infeasible. Instead, the influence function relies on a heavier approximation to estimate the effect of perturbing a single training point, rather than actually forgetting the training samples. In contrast, our approach only requires running the unlearning algorithm *once* for a given synthesized image query. This allows us to use a more mild approximation and obtain a model that forgets the synthesized sample. Guo *et al.* [20] and Tanno *et al.* [21] explore a similar formulation for unlearning training images and draw a connection between their unlearning algorithms and influence function. Our approach aims to unlearn the synthesized image instead, which connects to influence function in a similar fashion. We discuss our method's connection to influence function in more detail in Appendix A.3.

Optimizing a subset of weights. To further regularize the unlearning process, we optimize a small subset of weights, specifically W^k and W^v, the key and value projection matrices in cross-attention layers [56, 57]. In text-to-image models, cross-attention facilitates text-to-image binding, where W^k identifies which features match each text token, while W^v determines how to modify the features for the matched patches. We find that performing unlearning W^k and W^v is effective for attribution. Prior works also select the same set of parameters to improve fine-tuning [23] and unlearning [36].
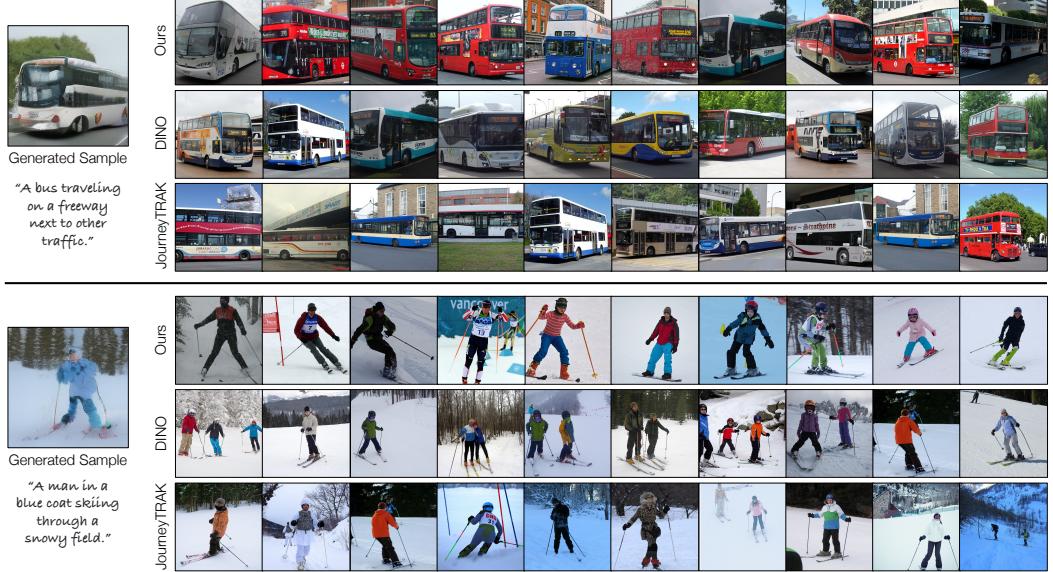
Figure 2: **Attribution results on MSCOCO models.** We show generated samples used as a query on the left, with training images being identified by different methods on the right. Qualitatively, our method retrieves images with more similar visual attributes. Notably, our method better matches the poses of the buses (considering random flips during training) and the poses and enumeration of skiers.

Implementation details. We conduct our studies on text-conditioned latent diffusion models [2]. Since diffusion models are typically trained with $T = 1000$ steps, evaluating the loss for all timesteps is costly. Therefore, we speed up computation by calculating the loss $\mathcal{L}(\mathbf{z}, \theta)$ with strided timesteps; we find that using a stride of 50 or 100 leads to good attribution performance. For calculating the loss change $\Delta \mathcal{L}(\hat{\mathbf{z}}, \theta)$ during evaluation, we take a finer stride of 5 steps to ensure a more accurate estimation of the DDPM loss. Additional details of our method, including hyperparameter choices, are provided in Appendix B.

5   Experiments

We validate our method in two ways. The first is a reliable, "gold-standard", but intensive – retraining a model from scratch without influential images identified by the algorithm. In Section 5.1, we perform this evaluation on a medium-sized dataset of 100k MSCOCO images [22]. Secondly, in Section 5.2, we evaluate our method on the Customized Model Benchmark [18], which measures attribution through customization on Stable Diffusion models [2]. This tests how well our method can apply to large-scale text-to-image models.

5.1   Leave-K-out counterfactual evaluation

Evaluation protocol. We select latent diffusion models [2] trained on MSCOCO [22], as its moderate size (118,287 images) allows for repeated leave-K-out retraining. Specifically, we use the pre-trained model evaluated in Georgiev *et al.* [12]. As outlined in Section 3, for each synthesized image $\hat{\mathbf{z}}$, we measure the leave-K-out model's **(1) loss change** $\Delta \mathcal{L}(\hat{\mathbf{z}}, \theta)$ and **(2) deviation of generation** $\Delta G_\theta(\epsilon, \mathbf{c})$. The deviation is measured by mean square error (MSE) and CLIP similarity [53]. We collect 110 synthesized images from the pre-trained model for evaluation, with different text prompts sourced from the MSCOCO validation set. We evaluate $\Delta \mathcal{L}(\hat{\mathbf{z}}, \theta)$ and $\Delta G_\theta(\epsilon, \mathbf{c})$ for all synthesized images and report mean and standard error.

We compare our method with several baselines:

- **Image similarity:** pixel space, CLIP image features [53], DINO [17], and DINOv2 [58]
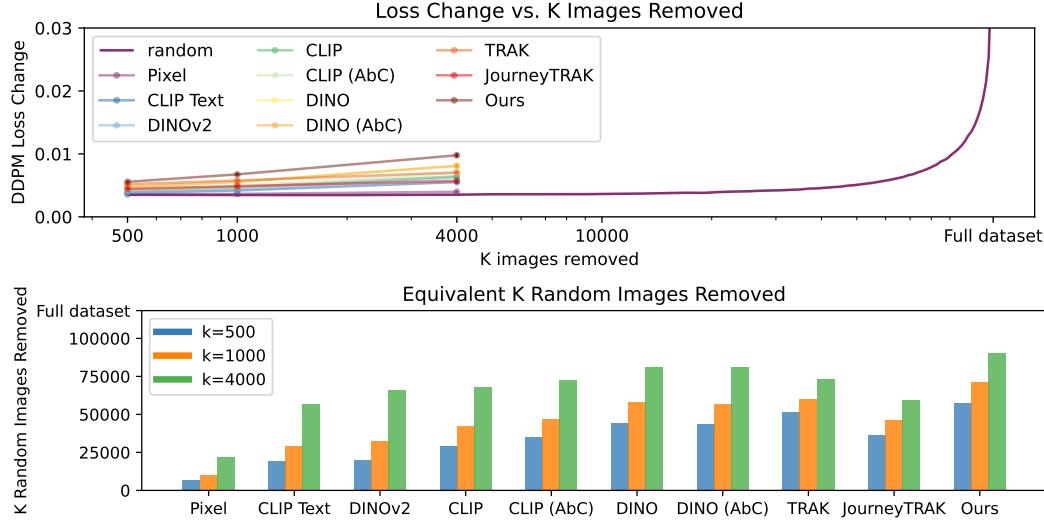- **Text similarity:** CLIP text features

6

Figure 3: **Evaluating loss changes in leave-K-out models.** Given a synthesized image $\hat{z}$, we train leave-K-out models for each of the attribution methods and track $\Delta\mathcal{L}(\hat{z}, \theta)$, the increase in loss change. **(Top)** We plot DDPM loss change versus K. The loss change becomes more significant with a larger K value, and our method (brown) yields the largest loss change for every K, indicating our method more successfully identifies the influential images. The purple curve represents models where random images are removed, serving as a reference. We show error bars representing 1 standard error in this plot (which are small and negligible). **(Bottom)** For each method, we show the equivalent number of random images needed to achieve a loss change.

- **Attribution by Customization [18] (AbC):** fine-tuned image features trained on the Customized Model benchmark, denoted as CLIP (AbC) and DINO (AbC)
- **Influence function:** Both TRAK and JourneyTRAK [12] are influence function-based methods that match the loss gradients of training and synthesized images, using random projection for efficiency. Both methods run the influence function on multiple models trained on the same dataset (20 in this test) and average the scores. The main difference is in the diffusion loss calculation: TRAK randomly samples and averages the loss over timesteps, while JourneyTRAK calculates it only at $t = 400$ for synthesized images during counterfactual evaluation.
- **Random:** We train models with K random images removed, using 10 models per value of K, sweeping through $K = 500, 1000, 2000, 3000, 4000$, and then 5000 to the full dataset size by increments of 2000.

For attribution methods, we use $K = 500, 1000$, and 4000, representing approximately 0.42%, 0.85%, and 3.4% of the MSCOCO dataset, respectively. Densely sweeping K is generally not feasible for attribution methods, as each queried synthesized image requires retraining a different set of K images. To provide better intuition, we report the number of *random* images that need to be removed from the dataset to achieve the same forgetting effect as removing K images from an attribution method.

Visual comparison of attributed images. In Figure 2, we find that our method, along with other baselines, can attribute synthesized images to visually similar training images. However, our method more consistently attributes images with the same fine-grained attributes, such as object location, pose, and counts. We provide more results in Appendix C.1. Next, we proceed with the counterfactual analysis, where we test whether these attributed images are truly *influential*.

Tracking loss changes in leave-K-out models. First, we report the change in DDPM loss for leave-K-out models in Figure 3. Matching in plain pixel or text feature space yields weak performance, while deep image features, particularly DINO, perform better. Interestingly, DINO outperforms influence function methods at $K = 4000$, despite not being trained specifically for the attribution task. Fine-tuning image features with the Customized Model benchmark, such as CLIP (AbC), shows
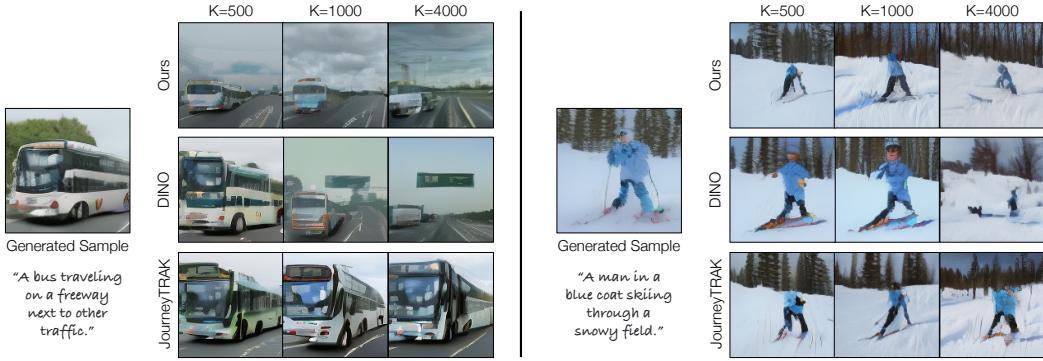
7

Figure 4: **Leave-K-out analysis for MSCOCO models.** We compare images across our method and baselines generated by leave-K-out models, using different K values, all under the same random noise and text prompt. A significant deviation in regeneration indicates that critical, influential images were identified by the attribution algorithm. While baselines regenerate similar images to the original, our method generates ones that deviate significantly, even with as few as 500 influential images removed ($\sim 0.42\%$ of the dataset).
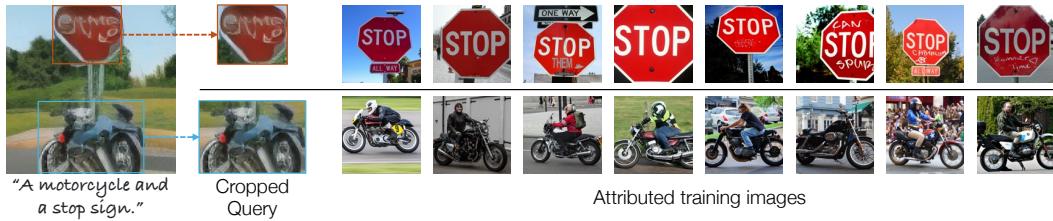

Figure 5: **Spatially-localized attribution.** Given a synthesized image (left), we crop regions containing specific objects using GroundingDINO [59]. We attribute each object separately by only running forgetting on the pixels within the cropped region. Our method can attribute different synthesized regions to different training images.

some improvement. However, in general, the improvement is limited, indicating that transferring from attributing customized models to general models remains challenging [18].

Among influence functions, TRAK significantly outperforms JourneyTRAK. We hypothesize that this is because JourneyTRAK collects gradients only for denoising loss at timestep $t = 400$, making it less effective for identifying influential images that affect the DDPM training loss different noise levels. Our method consistently performs best across all K values, outperforming both influence functions and feature-matching methods. For example, removing 500 images (just 0.42% of the dataset) using our method is equivalent to removing 57.5k random images (48.6% of the dataset) vs. 51.6k images (43.6%) and $\sim$44.3k images (37.5%) images for the best-performing baselines, TRAK and DINO.

Deviation of generated output in leave-K-out models. Figure 4 shows the deviation in generated outputs for leave-K-out models, where all images are generated using the same noise input and text prompt. Our method yields the largest deviation with a small budget. DINO is the second strongest method in this evaluation, where it also starts significantly altering the generated output with higher K. In contrast, influence functions, including TRAK and JourneyTRAK, perform subpar in this test, whereas JourneyTRAK outperforms TRAK in this context. We report quantitative results, including MSE and CLIP similarities of images between the pre-trained and leave-K-out models, along with ablation studies and more qualitative results, in Appendix C.1.

Spatially-localized attribution. While our formulation is written for whole images, we can run attribution on specific regions with little modification. We demonstrate this in Figure 5 on a generated image of a motorcycle and stop sign, using bounding boxes identified by GroundingDINO [59]. For each detected object, we run our unlearning (using the same prompt) on that specific object by optimizing the objective only within the bounding box. By doing so, we attribute different training images for the stop sign and motorcycle.
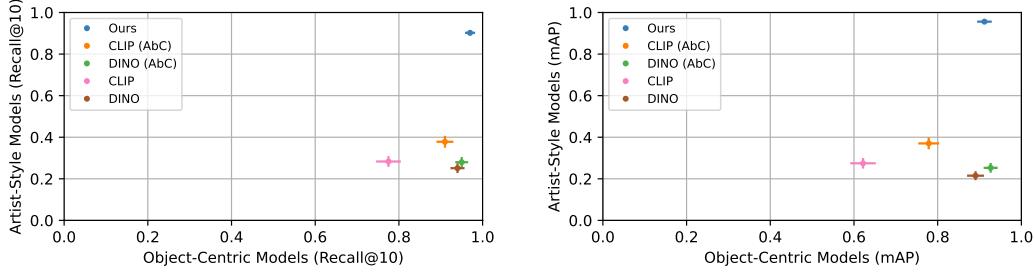
8

Figure 6: **Customized Model benchmark [18].** We report Recall@10 (left) and mAP (right) and show performance on artist-style models (y-axis) vs. object-centric models (x-axis). On object-style models, our method performs on par with AbC features, which were directly tuned on the benchmark, while significantly outperforming them on artist-style models. We plot 1 standard error on both axes.
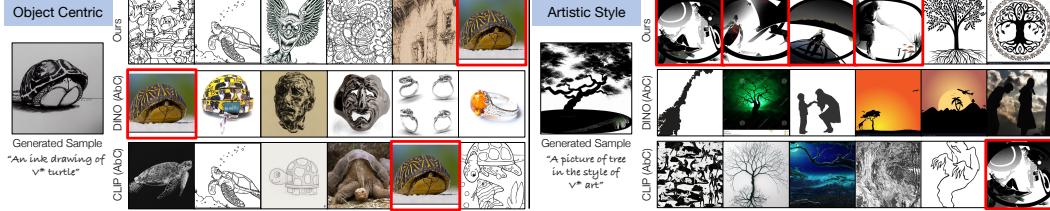

Figure 7: **Qualitative examples on the Customized Model benchmark.** The red boxes indicate ground truth exemplar images used for customizing the model. Both our method and baselines successfully identify the exemplar images on object-centric models (left), while our method outperforms the baselines with artistic style models (right).

5.2 Customized Model Benchmark

Wang *et al.* [18] focus on a specialized form of attribution: attributing customized models trained on an individual or a few exemplar images. This approach provides ground truth attribution since the images generated by customized models are computationally influenced by exemplar images. While this evaluation has limited generalization to attribution performance with larger training sets, it is the only tractable evaluation for attributing large-scale text-to-image models to date.

Evaluation protocol. Since the Customized Model Benchmark has ground truth, the problem is evaluated as a retrieval task. We report **Recall@K** and **mAP**, measuring the success of retrieving the exemplar images amongst a set including 100K LAION images. We compare with Wang *et al.*'s feature-matching approach that finetunes on the Customized Model dataset, referred to as DINO (AbC) and CLIP (AbC). For our evaluation, we selected a subset of the dataset comprising 20 models: 10 object-centric and 10 artist-style models. We select 20 synthesized images with different prompts for each model, resulting in 400 synthesized image queries.

Comparing with AbC features. We report Recall@10 and mAP in Figure 6. Our method performs on par with baselines when testing on object-centric models, while significantly outperforming them on artist-style models. Although CLIP (AbC) and DINO (AbC) are fine-tuned for this attribution task, the feature matching approach can sometimes confuse whether to attribute a synthesized image to style-related or object-related images. In contrast, our method, which has access to the model itself, traces influential training images more effectively. In Figure 7, we show a qualitative example. While DINO (AbC) and CLIP (AbC) can retrieve visually or semantically similar images, our method successfully identifies the exemplars in both cases. We include ablation studies in Appendix C.2.

6 Discussion, Broader Impacts, and Limitations

Generative models have entered the public consciousness, spawning companies and ecosystems that are deeply impacting the creative industry. The technology raises high-stakes ethical and legal questions surrounding the authorship of generated content [60, 61, 62, 63]. Data attribution is a critical piece of understanding the behavior of generative models, with potential applications in

9

informing a compensation model for rewarding contributors for training data. In addition, data attribution can join other works [64, 65, 66, 67] as a set of tools that allow end users to interpret how and why a model behaves, enabling a more trustworthy environment for machine learning models.

Our work proposes a method for data attribution for text-to-image models, leveraging model unlearning. We provide a counterfactual validation, verifying that removing the identified influential images indeed destroys the target image. While our method empirically demonstrates that unlearning can be effectively used, work remains to make this practical. Though our model unlearns efficiently, estimating the reconstruction loss on the training set remains a bottleneck, as several forward passes are required on each training estimate. While our evaluation showed that unlearning is useful for attribution, direct evaluation of unlearning algorithms for large generative models remains an open research challenge. Furthermore, to find a critical set of images, our method and baselines assign influence scores to individual images and sort them. However, groups of images may have interactions that are not captured in such a system. Furthermore, our method and baselines explore attribution of the whole image, while finer attribution on individual aspects of the image, such as style, structure, or individual segments, are of further interest.

References

[1] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[2] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[4] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[5] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Ho Jonathan, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[6] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. *arXiv preprint arXiv:2203.13131*, 2022.

[7] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[8] Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Patrick Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. In *International Conference on Machine Learning (ICML)*, 2023.

[9] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[10] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

[11] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*, 2022.

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

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

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

[15] Frank R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 1974.

[16] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*, 2023.

[17] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

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

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13):3521–3526, 2017.

[20] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning (ICML)*, 2020.

[21] Ryutaro Tanno, Melanie F Pradier, Aditya Nori, and Yingzhen Li. Repairing neural networks by leaving the right past behind. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[23] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[24] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

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

[26] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[27] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

[28] Han Guo, Nazneen Fatema Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*, 2020.

[29] Lloyd S Shapley. *A value for n-person games*. Princeton University Press Princeton, 1953.

[30] Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

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

[32] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[33] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *International Conference on Machine Learning (ICML)*, 2022.

[34] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Conference on Neural Information Processing Systems (NeurIPS)*, 33:19920–19930, 2020.

[35] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *arXiv preprint arxiv:2208.12242*, 2022.

[36] Nupur Kumari, Bingliang Zhang, Sheng-Yu Wang, Eli Shechtman, Richard Zhang, and Jun-Yan Zhu. Ablating concepts in text-to-image diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[37] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[38] Rohit Gandikota, Hadas Orgad, Yonatan Belinkov, Joanna Materzyńska, and David Bau. Unified concept editing in diffusion models. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

[39] Chi-Pin Huang, Kai-Po Chang, Chung-Ting Tsai, Yung-Hsuan Lai, and Yu-Chiang Frank Wang. Receler: Reliable concept erasing of text-to-image diffusion models via lightweight erasers. *arXiv preprint arXiv:2311.17717*, 2023.

[40] Eric Zhang, Kai Wang, Xingqian Xu, Zhangyang Wang, and Humphrey Shi. Forget-me-not: Learning to forget in text-to-image diffusion models. *arXiv preprint arXiv:2303.17591*, 2023.

[41] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.

[42] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Conference on Artificial Intelligence (AAAI)*, 2021.

[43] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[44] Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[45] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[46] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[47] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[48] Yongjing Zhang, Zhaobo Lu, Feng Zhang, Hao Wang, and Shaojing Li. Machine unlearning by reversing the continual learning. *Applied Sciences*, 2023.

[49] Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Conference on Artificial Intelligence (AAAI)*, 2024.

[50] Xi Shen, Alexei A Efros, and Mathieu Aubry. Discovering visual patterns in art collections with spatially-consistent feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9278–9287, 2019.

[51] Xuan Su, Jiaming Song, Chenlin Meng, and Stefano Ermon. Dual diffusion implicit bridges for image-to-image translation. In *International Conference on Learning Representations (ICLR)*, 2023.

[52] Valentin Khrulkov, Gleb Ryzhakov, Andrei Chertkov, and Ivan Oseledets. Understanding ddpm latent codes through optimal transport. In *International Conference on Learning Representations (ICLR)*, 2023.

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

[54] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[55] Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[56] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[58] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[59] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[60] Katherine Lee, A Feder Cooper, and James Grimmelmann. Talkin"bout ai generation: Copyright and the generative-ai supply chain. *arXiv preprint arXiv:2309.08133*, 2023.

[61] Niva Elkin-Koren, Uri Hacohen, Roi Livni, and Shay Moran. Can copyright be reduced to privacy? *arXiv preprint arXiv:2305.14822*, 2023.

[62] Uri Hacohen, Adi Haviv, Shahar Sarfaty, Bruria Friedman, Niva Elkin-Koren, Roi Livni, and Amit H Bermano. Not all similarities are created equal: Leveraging data-driven biases to inform genai copyright disputes. *arXiv preprint arXiv:2403.17691*, 2024.

[63] Ziv Epstein, Aaron Hertzmann, Hany Farid, Jessica Fjeld, Morgan R. Frank, Matthew Groh, Laura Herman, Neil Leach, Robert Mahari, Alex "Sandy" Pentland, Olga Russakovsky, Hope Schroeder, and Amy Smith. Art and the science of generative ai. *Science*, 380(6650):1110–1111, 2023.

[64] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Zhou Bolei, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[65] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual knowledge in gpt. *arXiv preprint arXiv:2202.05262*, 2022.

[66] Alex Foote, Neel Nanda, Esben Kran, Ioannis Konstas, Shay Cohen, and Fazl Barez. Neuron to graph: Interpreting language model neurons at scale. *arXiv preprint arXiv:2305.19911*, 2023.

[67] Amil Dravid, Yossi Gandelsman, Alexei A Efros, and Assaf Shocher. Rosetta neurons: Mining the common units in a model zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

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

[69] Dan Ruta, Saeid Motiian, Baldo Faieta, Zhe Lin, Hailin Jin, Alex Filipkowski, Andrew Gilbert, and John Collomosse. Aladin: All layer adaptive instance normalization for fine-grained style similarity. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[70] Mark Harden. Mark harden's artchive. `https://www.artchive.com/`, 1998.

A Derivations for Unlearning

In Section 4, we describe our unlearning method and its relationship to influence functions. Here, we provide more detailed derivations. Let θ_0 be the pretrained model trained on dataset $\mathcal{D}$ and loss $\sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta)$. N is the size of the dataset. Our goal is to obtain a model $\theta_{-\hat{\mathbf{z}}}$ that unlearns the synthesized image $\hat{\mathbf{z}}$.

A.1 EWC Loss

We summarize EWC Loss [19], which is the second order Taylor approximation of the data loss $\sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta)$ around θ_0. We denote the Hessian of the loss as H_{θ_0}, where $[H_{\theta_0}]_{ij} = \frac{1}{N} \frac{\partial^2}{\partial \theta_i \partial \theta_j} \sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta)|_{\theta = \theta_0}$. We denote the remainder term as $R(\theta)$.

$$
\begin{aligned}
\sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta) &= \sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta_0) + \sum_{\mathbf{z} \in \mathcal{D}} \nabla \mathcal{L}(\mathbf{z}, \theta)|_{\theta = \theta_0} (\theta - \theta_0) + \frac{N}{2} (\theta - \theta_0)^T H_{\theta_0} (\theta - \theta_0) + R(\theta) \\
&\approx \sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta_0) + \frac{N}{2} (\theta - \theta_0)^T H_{\theta_0} (\theta - \theta_0).
\end{aligned}
\tag{7}
$$

We assume that pretrained θ_0 is near the local minimum of the loss, resulting in a near-zero gradient $\sum_{\mathbf{z} \in \mathcal{D}} \nabla \mathcal{L}(\mathbf{z}, \theta_0)|_{\theta = \theta_0}$. We drop out higher order remainder term $R(\theta)$ in our approximation. If the model is trained with a negative log-likelihood loss, the Hessian H_{θ_0} is equivalent to the Fisher information F [27], leading to:

$$
\sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta) \approx \sum_{\mathbf{z} \in \mathcal{D}} \mathcal{L}(\mathbf{z}, \theta_0) + \frac{N}{2} (\theta - \theta_0)^T F (\theta - \theta_0).
\tag{8}
$$

We note that we focus on diffusion models, which are trained on a lower bound of the log-likelihood. In this context, the Fisher information can be viewed as the Gauss-Newton approximation of the Hessian [68]. The formulation satisfies the purpose of approximating the training loss on the dataset and serves as an effective regularizer on our unlearning objective.

Diagonal approximation of Fisher. Since Fisher information is the covariance of the log-likelihood gradient, its diagonal approximation is equivalent to taking the square of the gradients and averaging them across the training set. This diagonal approximation is adopted by EWC loss [19, 55]. In the context of diffusion models, the Fisher information is estimated by averaging across training data, random noise, and timesteps.

A.2 Updating step for unlearning

We then derive the Newton update of our unlearning objective in Equation 5. Below, we repeat our unlearning objective in Equation 4:

$$
\mathcal{L}_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta) \approx -\mathcal{L}(\hat{\mathbf{z}}, \theta) + \frac{N}{2} (\theta - \theta_0)^T F (\theta - \theta_0).
\tag{9}
$$

The Newton step is a second-order update of the form below, where α controls the step size:

$$
\theta \leftarrow \theta - \alpha \left[H_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta) \right]^{-1} \nabla \mathcal{L}_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta),
\tag{10}
$$

where $H_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta)$ is the Hessian of $\mathcal{L}_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta)$. Now we derive $\nabla \mathcal{L}_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta)$:

$$
\begin{aligned}
\nabla \mathcal{L}_{\text{unlearn}}^{\hat{\mathbf{z}}}(\theta) &\approx -\nabla \mathcal{L}(\hat{\mathbf{z}}, \theta) + N \cdot F (\theta - \theta_0) \\
&\approx -\nabla \mathcal{L}(\hat{\mathbf{z}}, \theta),
\end{aligned}
\tag{11}
$$

where we assume θ is close to θ_0, so the term $N \cdot F(\theta - \theta_0)$ can be omitted. Empirically, we also tried unlearning with this term added, but observed little change in performance. Then, we derive $H^{\hat{\mathbf{z}}}_{\text{unlearn}}(\theta)$ as follows:

$$
\begin{aligned}
H^{\hat{\mathbf{z}}}_{\text{unlearn}}(\theta) &\approx -H_{\hat{\mathbf{z}}}(\theta) + N \cdot F \\
&\approx N \cdot F,
\end{aligned}
\tag{12}
$$

where $H_{\hat{\mathbf{z}}}(\theta)$ is the Hessian of $\mathcal{L}(\hat{\mathbf{z}}, \theta)$. We assume the magnitude (in Forbenius norm) of $H_{\hat{\mathbf{z}}}(\theta)$ is bounded, and with a large dataset size N, we can approximate the Hessian $H^{\hat{\mathbf{z}}}_{\text{unlearn}}(\theta)$ as $N \cdot F$ only. Incorporating Equation 11 and 12 into Equation 10, we obtain our Newton update in Equation 5:

$$
\theta \leftarrow \theta + \frac{\alpha}{N} F^{-1} \nabla \mathcal{L}(\hat{\mathbf{z}}, \theta).
\tag{13}
$$

A.3 Connection to influence functions

We note that a special case of our formulation, running our update step once, with a small step size, is close to the formulation of influence functions. The difference is mainly on the linear approximation error of the loss on the training point. Starting with pretrained model θ_0 and taking an infinitesimal step γ:

$$
\theta_{-\hat{\mathbf{z}}} = \theta_0 + \gamma F^{-1} \nabla \mathcal{L}(\hat{\mathbf{z}}, \theta).
\tag{14}
$$

When we evaluate the loss of the training point $\mathbf{z}$ using the unlearned model $\theta_{-\hat{\mathbf{z}}}$, we can write the loss in a linearized form around θ_0, as we taking a small step:

$$
\begin{aligned}
\mathcal{L}(\mathbf{z}, \theta_{-\hat{\mathbf{z}}}) &\approx \mathcal{L}(\mathbf{z}, \theta_0) + \nabla \mathcal{L}(\mathbf{z}, \theta_0)(\theta_{-\hat{\mathbf{z}}} - \theta_0) \\
&= \mathcal{L}(\mathbf{z}, \theta_0) + \gamma \nabla \mathcal{L}(\mathbf{z}, \theta_0) F^{-1} \nabla \mathcal{L}(\hat{\mathbf{z}}, \theta).
\end{aligned}
\tag{15}
$$

Now, we plug Equation 15 into our attribution function in Equation 6:

$$
\begin{aligned}
\tau(\hat{\mathbf{z}}, \mathbf{z}) &= \mathcal{L}(\mathbf{z}, \theta_{-\hat{\mathbf{z}}}) - \mathcal{L}(\mathbf{z}, \theta_0) \\
&\approx \gamma \nabla \mathcal{L}(\mathbf{z}, \theta_0) F^{-1} \nabla \mathcal{L}(\hat{\mathbf{z}}, \theta).
\end{aligned}
\tag{16}
$$

In this special case, our method is equivalent to influence function $\nabla \mathcal{L}(\mathbf{z}, \theta_0) F^{-1} \nabla \mathcal{L}(\hat{\mathbf{z}}, \theta)$, after approximations. Practically, the difference between our method and influence functions is that we are taking larger, sometimes multiple steps (rather than a single, infinitesimal step), and are explicitly evaluating the loss (rather than with a linear, closed-form approximation).

B Implementation Details

B.1 MSCOCO Models

Models trained from scratch. We select our source model for attribution from Georgiev *et al.* [12], which is a latent diffusion model where the CLIP text encoder and VAE are exactly the ones used in Stable Diffusion v2, but with a smaller U-Net. To retrain each MSCOCO model for leave-K-out evaluation, we follow the same training recipe as the source model, where each model is trained with 200 epochs, a learning rate of 10^{-4}, and a batch size of 128. We use the COCO 2017 training split as our training set.

Unlearning. To unlearn a synthesized sample in MSCOCO models, we find that running with 1 step already yields good attribution performance. We perform Newton unlearning updates with step sizes of 0.01 and update only cross-attention KV (W^k, W^v). We find that updating cross-attention KV yields the best performance, and we later provide ablation studies on the optimal subset of layers to update. We sample gradients 591,435 times to estimate the diagonal Fisher information, equivalent to 5 epochs of MSCOCO training set.

B.2 Customized Model Benchmark

Model collection. As described in Section 5.2, we selected a subset of the dataset [18] comprising of 20 models: 10 object-centric and 10 artist-style models. For all object-centric models, we select models with distinct categories. For artist-style models, we select 5 models trained from BAM-FG [69] exemplars and 5 models trained from Artchive [70] exemplars. To speed up computation, we calculate Fisher information on Stable Diffusion v1.4, the base model of all the customized models, over the selected subset of LAION images. We then apply the same Fisher information to all customized models.

Unlearning. We find that running 100 unlearning steps yields a much better performance than running with 1 step for this task. Moreover, updating only cross-attention KV yields a significant boost in performance in this test case. In Appendix C.2, we show an ablation study on these design choices. We sample gradients 1,000,000 times to estimate the diagonal Fisher information, where the gradients are calculated from the 100k Laion subset using Stable Diffusion v1.4.

B.3 Baselines.

Pixel space. Following JourneyTRAK's implementation [12], we flatten the pixel intensities and use cosine similarity for attribution.

CLIP image and text features. We use the official ViT-B/32 model for image and text features.

DINO. We use the official ViT-B/16 model for image features.

DINOv2. We use the official ViT-L14 model with registers for image features.

CLIP (AbC) and DINO (AbC). We use the official models trained on the combination of object-centric and style-centric customized images. CLIP (AbC) and DINO (AbC) are selected because they are the best-performing choices of features.

TRAK and Journey TRAK. We adopt the official implementation of TRAK and JourneyTRAK and use a random projection dimension of 4096, the same as what they use for MSCOCO experiments.

B.4 Additional Details

Horizontal flips. Text-to-image models in our experiments are all trained with horizontal flips. As a result, the models are effectively also trained with the flipped version of the dataset. Therefore, we run an attribution algorithm for each training image on its original and flipped version and obtain the final score by taking the max of the two. For a fair comparison, we adopt this approach for all methods. We also find that taking the average instead of the max empirically yields similar performance.

Computational resources. We conduct all of our experiments on A100 GPUs. It takes around 16 hours to train an MSCOCO model from scratch, 20 hours to evaluate all training image loss, and 2 minutes to unlearn a synthesized image from a pretrained MSCOCO model. To finish all experiments on MSCOCO models, it takes around 77K GPU hours. For Customized Model Benchmark, it takes 2 hours to unlearn a synthesized image and 16 hours to track the training image loss. To finish all experiments on this benchmark, it takes around 36K GPU hours.

Licenses. The source model from Georgiev *et al.* [12] (JourneyTRAK) is released under the MIT License. The MSCOCO dataset is released under the Creative Commons Attribution 4.0 License. Stable Diffusion v2 is released under the CreativeML Open RAIL++-M License. The CLIP model is released under the MIT License. The Customized Model Benchmark is released under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

C Additional Analysis

C.1 MSCOCO Models

Deviation of generated output in leave-K-out models. We report the quantitative evaluation for the deviation of generated output in terms of MSE in Figure 8 and in terms of CLIP similarity in Figure 9. We find that in terms of this metric, our method still outperforms all baselines while being slightly less performant than $K = 4000$ when compared with DINO features under CLIP
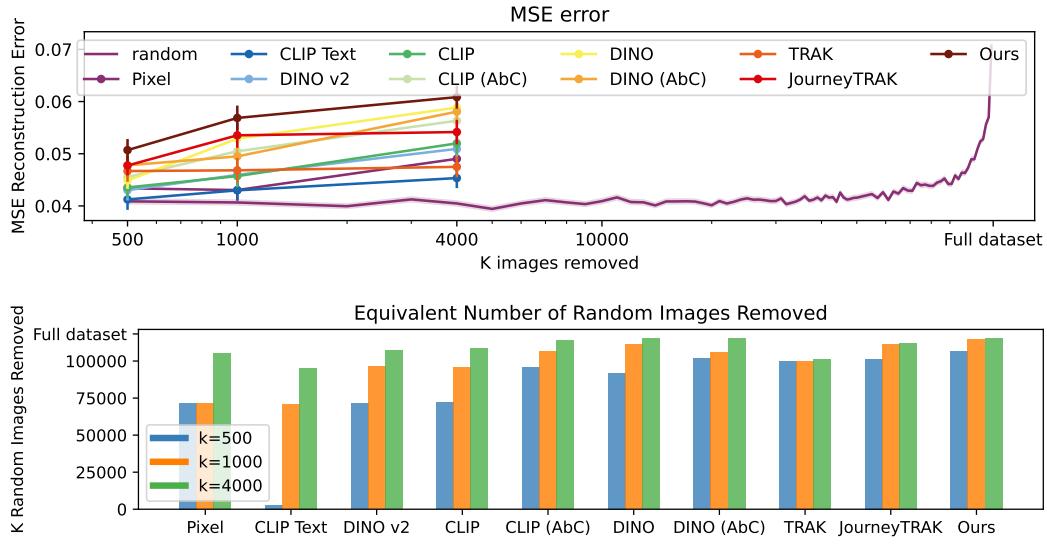
Figure 8: **Deviation of generated output in leave-K-out models, in mean square error (MSE).** We report the deviation of generated output in the same fashion as in Figure 3. The higher the MSE, the better since more deviation indicates that the K images removed are more influential.
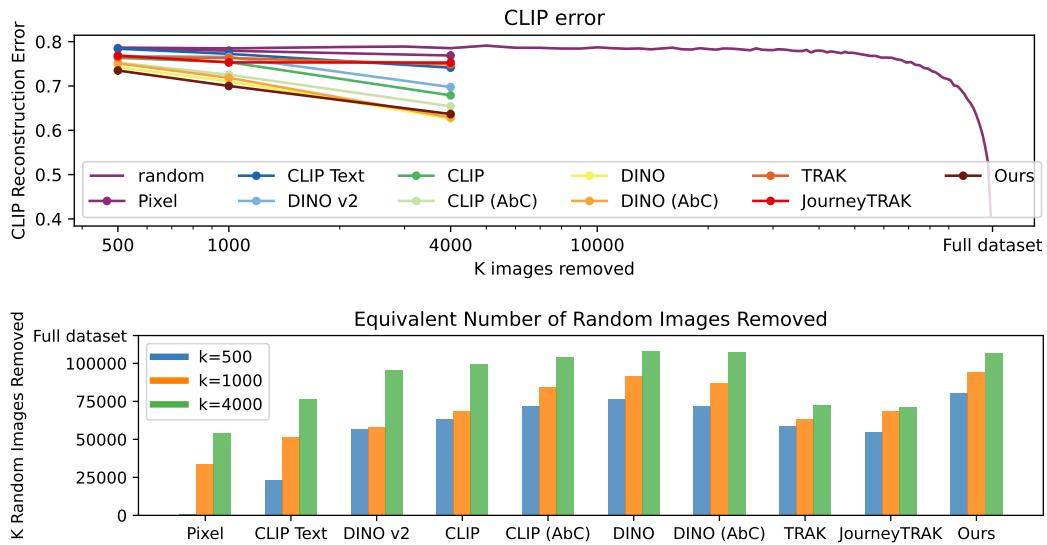

Figure 9: **Deviation of generated output in leave-K-out models, in CLIP similarity.** We report the deviation of generated output in the same fashion as in Figure 3. A lower CLIP similarity in leave-K-out models indicates a better attribution algorithm.

similarity. Interestingly, we find that JourneyTRAK performs better than TRAK in this metric despite significantly underperforming regarding loss changes (Figure 3). We note that DINO, despite not being designed for attribution, clearly outperforms both TRAK and JourneyTRAK in this evaluation setup.

Ablation studies. We perform the following ablation studies and select hyperparameters for best performance in each test case:

- **SGD (1 step):** 1 SGD step, step size 0.001
- **SGD (10 steps):** 10 SGD steps, step size 0.0001
- **Full weight:** 1 Newton steps, step size 0.0005
- **Attention:** 1 Newton steps, step size 0.005
- **Cross-attention:** 1 Newton steps, step size 0.005
- **Cross-attention KV:** 1 Newton steps, step size 0.01 (This is our final method)

The SGD step refers to the baseline of directly maximizing synthesized image loss without EWC loss regularization, as described in Section 4.

We also compare different subsets of weights to optimize and report loss change, deviation measured by MSE, and deviation measured by CLIP similarity in Figure 12, 13, 14, respectively. We find that the 4 choices of weight subset selection all lead to effective attribution performance, where restricting weight updates to cross-attention KV yields the best performance overall. However, both configurations for SGD updates perform much worse, indicating the importance of regulating unlearning with Fisher information.

Additional results. We provide more attribution results in Figure 10 and more results on leave-K-out models in Figure 11.

C.2 Customized Model Benchmark

Ablation studies. We perform the following ablation studies and select hyperparameters for best performance in each test case:

- **Cross-attention KV (100 steps):** 100 Newton steps, step size 0.1 (denoted as **Ours** in Figure 15)
- **Cross-attention KV (1 step):** 1 Newton step, step size 10
- **Full weight (100 steps):** 100 Newton steps, step size 5×10^{-5}
- **Cross-attention KV, SGD step (100 steps):** 100 SGD step, step size 0.01

Again, the SGD step refers to the baseline of directly maximizing synthesized image loss without EWC loss regularization, as described in Section 4.

We report the result of our ablation studies in Figure 15. Our findings indicate that for this test case, selecting a small subset of weights (i.e., cross-attention KV) combined with multiple unlearning steps (100 steps) is crucial for effective attribution. We hypothesize that stronger regularization is necessary for unlearning in larger-scale models, and that such models benefit more from numerous smaller unlearning steps rather than fewer, larger steps to achieve a better optimization.

Customized models in this benchmark associate the exemplar with a special token V^*, which is also used for generating synthesized images. Our method involves forgetting the synthesized image associated with its text prompt, so by default, we tested it with V^* included. Meanwhile, we also evaluated our method without V^* in the prompts. Figure 6 shows that removing V^* reduces performance, but the method still performs well overall.
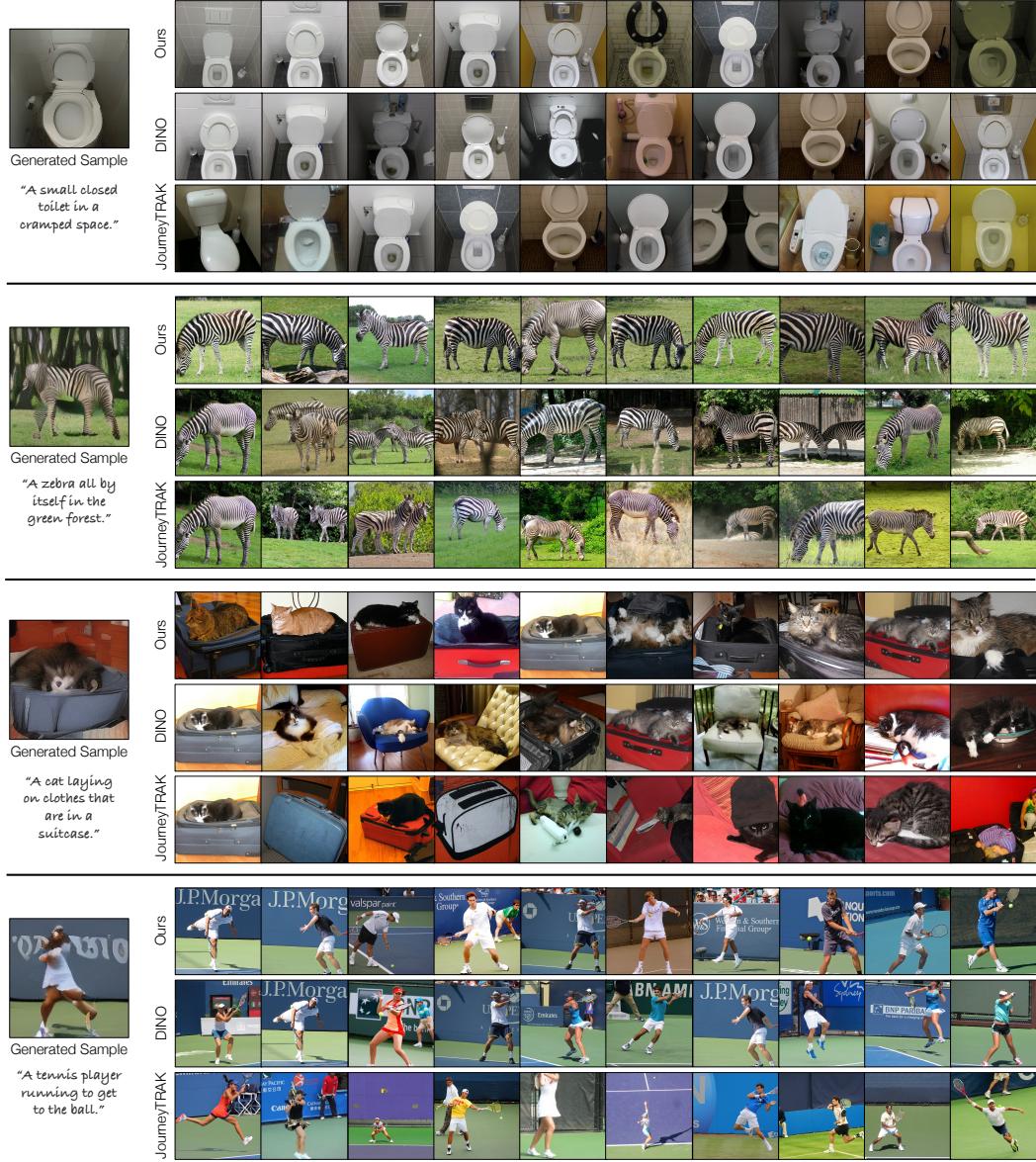
Figure 10: **Additional attribution results for MSCOCO models.** This is an extension of Figure 2 in the main paper. Our method identifies images with similar poses and visual attributions to the query image. Importantly, in Figure 11, we verify that leaving these images out of training corrupts the ability of the model to synthesize the queried images.

Figure 11: **Additional leave-K-out model results for MSCOCO models.** This is an extension of Figure 4 in the main paper, showing the results from removing top-K influential images from different algorithms, retraining, and attempting to regenerate a synthesized sample. The influential images for these examples are shown in Figure 10. Our method consistently destroys the synthesized examples, verifying that our method is identifying the critical influential images.

Figure 12: **Ablation studies for attributing MSCOCO models.** We report the change in synthesized image loss in leave-K-out models in the same fashion as in Figure 3 in the main paper. We compare four different sets of weights to unlearn (full, attention layers, cross-attention layers, and cross-attention W^k, W^v), and we find that cross-attention W^k, W^v outperforms other configurations. We also evaluate a naive variation where we apply SGD to maximize the synthesized image's loss, and we find that updating with 1 steps or 10 steps both perform only at chance (random baseline).
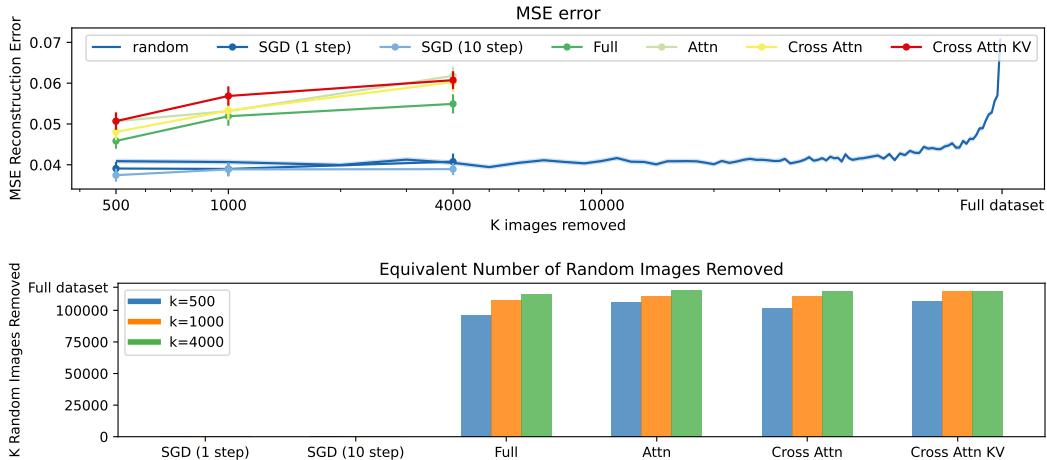
Figure 13: **Ablation studies for attributing MSCOCO models.** We report the deviation of generated output in leave-K-out models in mean square error (MSE) in the same fashion as in Figure 8. We find that the trend of each ablation follows that of Figure 12.
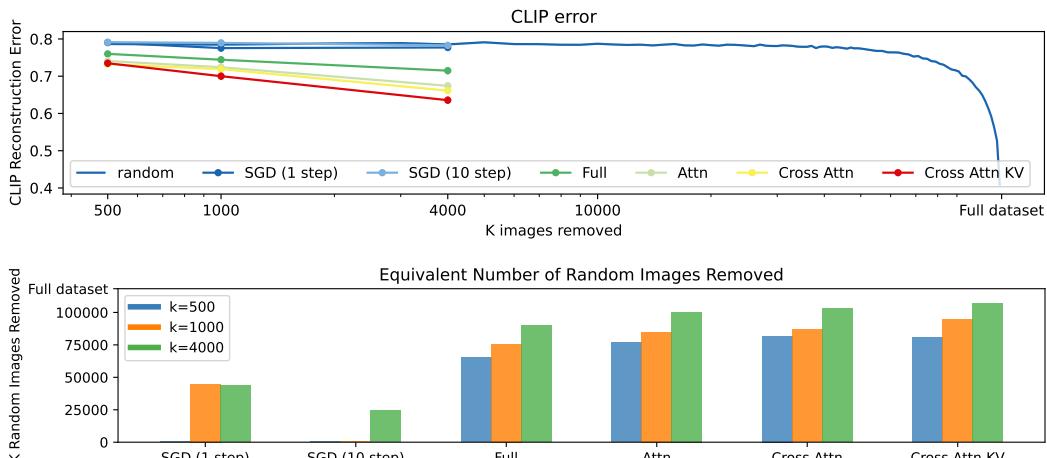
.

Figure 14: **Ablation studies for attributing MSCOCO models.** We report the deviation of generated output in leave-K-out models in CLIP similarity in the same fashion as in Figure 9. We find that the trend of each ablation follows that of Figure 12.
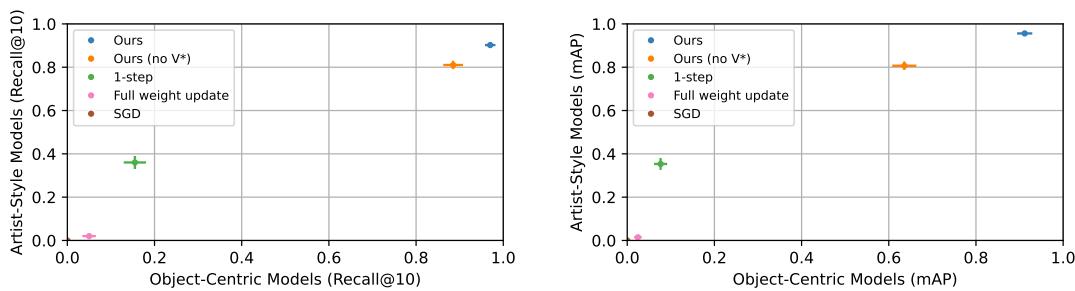

Figure 15: **Ablation studies for Customized Model Benchmark.** We report evaluation on the Customized Model Benchmark in the same fashion as in Figure 6. We find that training with multiple steps, updating a selected subset of weights, and regularizing unlearning via Fisher information is crucial to this task. Additionally, we test a version where we apply our algorithm without the special token V^*. While it reduces performance, it still performs well in overall.