PNAS PERSPECTIVE OPEN ACCESS

# AI model disgorgement: Methods and choices

Alessandro Achille[a], Michael Kearns[a,b,1], Carson Klingenberg[a], and Stefano Soatto[a,c]

Edited by Jeffrey Ullman, Stanford University (Retired), Stanford, CA; received July 12, 2023; accepted March 15, 2024

**Over the past few years, machine learning models have significantly increased in size and complexity, especially in the area of generative AI such as large language models. These models require massive amounts of data and compute capacity to train, to the extent that concerns over the training data (such as protected or private content) cannot be practically addressed by retraining the model "from scratch" with the questionable data removed or altered. Furthermore, despite significant efforts and controls dedicated to ensuring that training corpora are properly curated and composed, the sheer volume required makes it infeasible to manually inspect each datum comprising a training corpus. One potential approach to training corpus data defects is model disgorgement, by which we broadly mean the elimination or reduction of not only any improperly used data, but also the effects of improperly used data on any component of an ML model. Model disgorgement techniques can be used to address a wide range of issues, such as reducing bias or toxicity, increasing fidelity, and ensuring responsible use of intellectual property. In this paper, we survey the landscape of model disgorgement methods and introduce a taxonomy of disgorgement techniques that are applicable to modern ML systems. In particular, we investigate the various meanings of "removing the effects" of data on the trained model in a way that does not require retraining from scratch.**

machine learning | artificial intelligence | model disgorgement | machine unlearning | generative AI

Responsible use of data is an indispensable part of any machine learning (ML) implementation. ML developers must carefully collect and curate their datasets, and document their provenance. They must also make sure to respect intellectual property rights, preserve individual privacy, and use data in an ethical way. Consider in particular the following concerns related to the training of ML models—especially generative AI models such as large language models (LLMs) or diffusion models—on private, protected, or otherwise sensitive data:

- Personal data. The training data contains identifying information about individuals, such as physical addresses, workplace, or social media activity. Even if such data was posted on the open Internet or other public forums, for privacy reasons we (or the affected individuals) would prefer that it not be produced verbatim in the output of a model, either inadvertently or as the result of targeted model inputs (prompts).
- Proprietary data. The training data contains material that is subject to copyright or other intellectual property protections. While globally there are a wide variety of protections for the usage of copyrighted material for machine learning use cases (for example, "fair use" in the United States and the "text and data mining" justification in the European Union), model developers still may uncover data defects related to intellectual property issues.
- Toxic data. The training data contains material that could be viewed as toxic, offensive, or otherwise inappropriate. A primary concern is that such data could influence model outputs to exhibit similarly undesirable behaviors.
- Low quality data. The training data may contain content that is badly annotated, noisy, or otherwise not useful or damaging for the model. We want to be able to remove such data when discovered to improve the model quality.

In each of the scenarios above, the underlying concerns are distinct, but in all of them, we desire to mitigate or eradicate the effects of the data mentioned on model outputs and behavior. However, due to significant increase in size and complexity over the past years, current ML models require a very large amount of data and compute capacity to train, to the extent that training corpus defects such as those listed above cannot be trivially remedied by retraining the model from scratch. Despite sophisticated controls on training data and a significant amount of effort dedicated to ensuring that training corpora are properly composed, the sheer volume of data required for the models makes it challenging to manually inspect each datum comprising a training corpus.

Thus one potentially more practical solution for training corpus data defects is model disgorgement*, by which informally we mean the reduction or elimination of the effects of improperly used data on any component of an ML model with minimal degradation in the model performance (utility). In this paper, we introduce a taxonomy of possible disgorgement methods that are applicable to modern ML systems. In particular, we investigate the meaning of "removing the effects" of data in the trained model in a way

Author affiliations: [a]Amazon Web Services Artificial Intelligence (AWS AI), Pasadena, CA 91125; [b]Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19130; and [c]Computer Science, University of California, Los Angeles, CA 90095

Author contributions: A.A., M.K., C.K., and S.S. wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission.



[1]To whom correspondence may be addressed. Email: mkearns@cis.upenn.edu.

Published April 19, 2024.

*The term "disgorgement" has specific meanings in legal contexts (1), as well as more recent usage in algorithmic and regulatory settings (2). Our use of the term "model disgorgement" is more closely related to the algorithmic context, and refers specifically to the disgorgement or mitigation of the effects of training data on machine learning models.

that does not require retraining from scratch. We shall not attempt to provide a precise and formal definition of model disgorgement, since there are a number of distinct and somewhat incomparable approaches and different fields have not yet converged to a single approach. Instead, we aim to informally cover the core established ideas and refer the reader to works for possible rigorous formalization.

Model disgorgement techniques ("MD" hereafter) can be used to address a wide range of issues, such as reducing bias or toxicity, increasing fidelity, and ensuring responsible usage of intellectual property. Developers of large models should understand their options for implementing model disgorgement, and novel methods to perform model disgorgement should be developed to help ensure the responsible usage of data.

This paper presents both novel and known technical approaches to MD. These approaches are distinguished along three dimensions:

1. Disgorgement guarantee. We must consider what it means to "remove the effects" of particular data on a model. Here, we draw a technical distinction between *deterministic* processes (where we can say that particular data has categorically not affected a model) versus probabilistic processes (where we can say that the effect of particular data on a model is effectively zero or de minimis in a statistical sense). While probabilistic guarantees are formalistically weaker, they may allow better utility of the disgorged model and, if properly implemented, can approximate the effectiveness of deterministic methods.
2. Disgorgement type. Here we make a distinction between methods for disgorgement that are structural and those that are behavioral. By structural we mean methods that alter the ML workflow structure in a way that enforces disgorgement, or facilitates later disgorgement. An example of the latter would be a model architecture that averages many smaller submodels, each trained on different partitions of the data; disgorgement of any particular portion of data is then achieved by dropping its corresponding submodel from the average. By behavioral we mean methods that modify the parameters or the training of the model to achieve approximate disgorgement, and for which the guarantees do not come from the structure of the method itself, but might require empirical measurement, or numerical quantification of the degree of disgorgement. Such methods include various "forgetting" or "unlearning" approaches as well as differential privacy.
3. Disgorgement temporal application. We also distinguish approaches based on when those techniques must be applied as part of the model construction process. We present approaches that can be applied reactively (applied post hoc to a fully trained model), proactively (where the possibility of later disgorgement must be explicitly anticipated as part of the training process), or preemptively (where the very nature of the training process minimizes the effects of any sufficiently small fraction of training data on the resulting model, thus obviating later explicit disgorgement actions).

The following table summarizes the techniques we will discuss, and their attributes in our proposed taxonomy. It should also be noted that the different methods are not mutually exclusive and may be used in combinations for example using both compartmentalization and differential privacy together (3).

| Disgorgement method | Guarantee | Type | Time |
|---|---|---|---|
| Retraining | Deterministic | Structural | Reactive |
| Forgetting/unlearning | Probabilistic | Empirical | Reactive |
| Compartmentalization | Deterministic | Structural | Proactive |
| Differential privacy | Probabilistic | Empirical | Preemptive |
| Dataset emulation | Deterministic | Structural | Preemptive |

## Organization and Contributions

The rest of the paper is structured as follows. In Sections 1, 2 and 3 we introduce, respectively, reactive, proactive and preemptive disgorgement methods, highlighting some prototypical examples from the literature. Previous literature in machine unlearning frequently casts under the same umbrella forgetting and compartmentalization methods, which however have different scopes; Sections 1 and 2 attempt to clarify the differences and explore the different trade-offs they allow. Similarly, practitioners sometimes improperly see differential privacy (Section 3) as a uniform protection for all data concerns. As we explore in Section 4, this is not the case and the full taxonomy of disgorgement should be used to offer proper solutions to each problem. Section 3 proposes dataset emulation, a relatively unexplored route to handle preemptive disgorgement in a deterministic fashion.

## 1. Reactive Disgorgement

Reactive disgorgement consists of all techniques that can be applied to a model after it has already been trained, in order to eliminate or reduce the effect of one or multiple samples. While the exact formalization of the objective varies, all methods attempt to modify the parameters (weights) of the model in order to selectively remove any information related to those samples, thus making the model indistinguishable from one trained without seeing that data to start with. Techniques for reactive disgorgement fall into two categories: retraining and selective forgetting.

**1.1. Retraining.** The conceptually simplest approach to realize reactive disgorgement is to completely discard the parameters of the model "contaminated" by the samples to disgorge, and train a new model from scratch on the remaining data. This however poses several practical challenges. First, current state-of-the-art neural networks have hundreds of billions of parameters, and retraining such a model requires a large resource investment. Hence, retraining the model in response to each MD request may not a viable approach, particularly if the system should be designed to handle a possible stream of small but frequent disgorgement requests. Checkpointing of the intermediate training stages of a model can reduce the retraining cost (4, 5), since retraining can start from the last checkpoint before the sample was seen. This is especially useful if samples more likely to require disgorgement can be

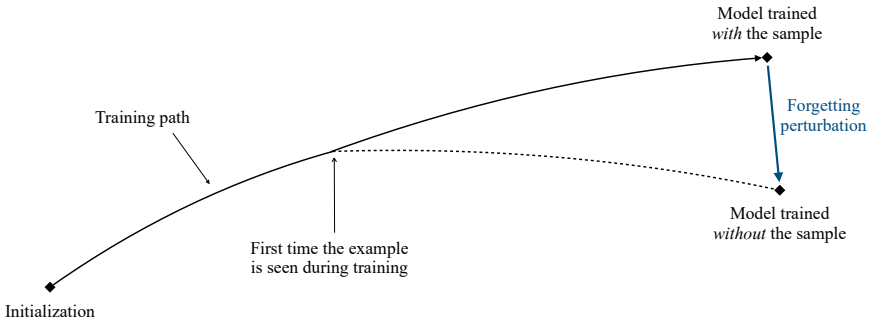


**Fig. 1.** Selective forgetting. The aim of disgorgement is to recover a model indistinguishable from one trained without some examples. Rather than training a new model from scratch, selective forgetting methods analyze the effect of the sample on the training dynamics of the model and apply a perturbation to the weights that remove the influence.

scheduled toward the end of the training. However, such prior information is often not available, and the worst-case cost of a forgetting request remains prohibitively large.

Apart from cost considerations, retraining poses challenging problems to end-users when the ML model is used as part of a workflow in a larger system. In particular, the behavior of the retrained model may generally be significantly different from the behavior of the original model (6, 7). In fact, slight perturbations of the training process, even if retraining the same model on the same data, can yield models whose behavior is sufficiently different to disrupt downstream workflows (8, 9). Thus, naive MD based on frequent retraining would render most large-scale ML systems essentially unusable.

**1.2. Selective Forgetting/Machine Unlearning.** Often the cohort of data that triggers the need for MD is a small fraction of the overall training set. In such cases, we expect its influence on the training model to be generally small and it may be possible to remove the influence by simply modifying the existing weights, rather than having to retrain from scratch. This approach is referred to in the literature as selective forgetting (10–12) or machine unlearning (13–18). See Fig. 1 for an illustration.[†]

Many forgetting algorithms are based on the notion of influence functions (19): Intuitively, being exposed to a sample during training slightly perturbs the training path taken by the model in parameter space (Section 1). If such perturbation is small, it may be possible to approximate it efficiently and directly predict what the weights would have been if the model had not seen the sample, without having to retrain the model. This procedure can be carried out exactly for simple models, such as ridge regression. However, exactly estimating the influence of a sample is highly challenging for complex models such as large-scale deep networks, due to the complexity of the training process and the highly nonconvex nature of the loss function (20).

Incorrect estimation of the influence potentially leads to information remaining in the weights of the model after forgetting. To account for this methods often add a low amount of noise to the weights of the network (10–12, 18), in order to erase potential remaining information. Due to the stochastic nature of the process, such methods cannot deterministically guarantee that no information is present in the weights, as was the case with retraining. Rather, they provide stochastic guarantees stating that, with high probability, the amount of information remaining in the system is less than a certain value that depends on the model, forgetting algorithm, and amount of noise added. For simpler models, such as models with a convex loss function, one can analytically compute the expected information remaining (11, 18) and thus offer strong guarantees. For more complex models, one must empirically measure the success of the forgetting procedure on a validation set (10, 12).

The addition of noise also highlights a fundamental trade-off between the cost of forgetting and utility of the model that is common to all disgorgement methods, whether stochastic or deterministic: Using a fast approximation of the influence of the sample reduces the computational cost of the forgetting procedure. However, more noise needs to be added to account for the expected error in estimating the sample influence. This noise in turn reduces the accuracy (utility) of the model. Retraining, on the other hand, is the most expensive disgorgement technique but results in the best utility. Practitioners should consider the trade-offs offered by the various methods and pick the best one for their application.

Measuring the success of the forgetting procedure—that is, the amount of remaining information—is also challenging (see Section 4 for more discussion). Some methods measure the difference between the parameters of the "scrubbed" model and the parameters of a golden reference model trained without the disgorged sample. This quantity, however, is difficult to convert to actionable user-facing metrics. Instead of comparing the parameters, other methods (10–12) measure the difference in behavior of the scrubbed and reference model, through readout functions such as the time that it takes to overfit the model on the disgorged data, or the confidence of the model (entropy of the predictions). On this note, it is important to note that forgetting a sample is not the same as having poor accuracy on that sample. For example, a model trained to distinguish "cats" and "dogs" should not necessarily misclassify a training image of a particular dog as a cat after forgetting it. Rather, it should output the same label as a model that was trained without that image. For this reason, methods that perform "selective degradation" (21–23)—that is, modify the model to perform

[†]"Machine unlearning" is however also often used to refer to other disgorgement methods that fall under compartmentalization (see Section 2). We use "selective forgetting" instead to keep the distinction clear.

worse or have less confidence on some examples—are not in general valid selective forgetting algorithms, although they may bring other benefits. In particular, practitioners have to be mindful of the Streisand effect (12): forcefully modifying the behavior of the model on a subset of samples may make them more visible and expose them to attacks.

Finally, we note that measuring the influence of samples is easier if those samples have been seen by the model only toward the end of the training. If possible then, one may want to initially train the model only on a safe subset of the data that is known to never require disgorgement (11), for instance, synthetic data generated ab ovo. This relates to proactive methods for disgorgement, which we discuss next.

## 2. Proactive Disgorgement

Given that even the tightest standards of data curation can be imperfect at the scale of the datasets in use today, it may be worthwhile to train ML models in preparation for possible MD. That is, the ML models could be trained in such a way that MD can be later performed with minimal impact on the overall trained model. We call this proactive disgorgement.

A conceptually simple and popular approach to proactive disgorgement is compartmentalization through a mixture of experts. Compartmentalization methods (4, 24–27) split the training dataset into multiple disjoint subsets (or shards) and train separate submodels (experts) in isolation on each shard. The submodels are then ensembled at inference time to obtain the final mixture of expert model. As a result, information (and influence) from different samples is separated into different submodels (hence the name "compartmentalization"), and disgorgement requests can be handled by eliminating or retraining only the components of the model that have been exposed to the cohort of data in question.

Compartmentalization brings multiple benefits. First, it lowers the computation cost of disgorgement, since only a subset of the parameters have to be retrained or disgorged. In fact, disgorgement can be accomplished without retraining, by simply deleting the affected submodels from the ensemble. Second, compartmentalization increases stability: since only a small subset of the components of the ensemble are modified, the behavior of the disgorged model will be close to the behavior of the original model, thus decreasing the risk of disrupting downstream workflows (Section 1). Third, inference-time algorithms can be designed that use the compartmentalized structure to remove the influence of individual samples from the model predictions, even when such influence is still present in the model parameters (28).

Splitting the data in increasing smaller shards leads to better disgorgement, since every disgorgement request will affect a smaller fraction of the total number of submodels. Moreover, the time to retrain each component depends on the size of its shard and will thus be lowered. On the other hand, using a finer sharding also has several downsides: 1) increased latency: the input needs to be forwarded through each model in the ensemble; 2) increased storage cost; 3) reduced utility: ensembling smaller models trained on small subsets of the data may lead to worse accuracy compared to a monolithic model trained on all the available data.

The main design choices for compartmentalized models thus concern how to split the training data, what architecture to use for the submodels, and how to ensemble them. These in turn affect three aspects of the trained ML system: expected forgetting cost, model utility, and inference time latency. Practitioners need to choose the desired trade-off between expected forgetting cost, accuracy and latency along a Pareto curve (Fig. 2).

Better model disgorgement design can lead to better Pareto curves or better trade-offs in the desired use cases. For example, ref. 4 proposes a baseline compartmentalization method that randomly splits the data into uniform shards and trains a separate copy of a standard network on each. On the other hand, ref. 24 starts with a network pretrained on a core dataset deemed safe from disgorgement requirements,[‡] and constructs simple linear adapters on the data by computing the average embedding. This leads to lower accuracy, but enables instant forgetting, where individual samples can be disgorged without any retraining. Combination of multiple strategies is also possible; for example, ref. 3 obtains uniformly better trade-off by a) optimizing the shard composition to improve the submodels' performance; b) using a similar mechanism to that of ref. 24 to ensure good accuracy for extremely low retraining time, and c) using expressive but fast adapters (29) as submodels, which allows training and running inference in parallel on thousands of shards with a significantly reduced computational cost. In a complementary fashion, (25) focuses on improving the ensembling procedure by making it instance-dependent.

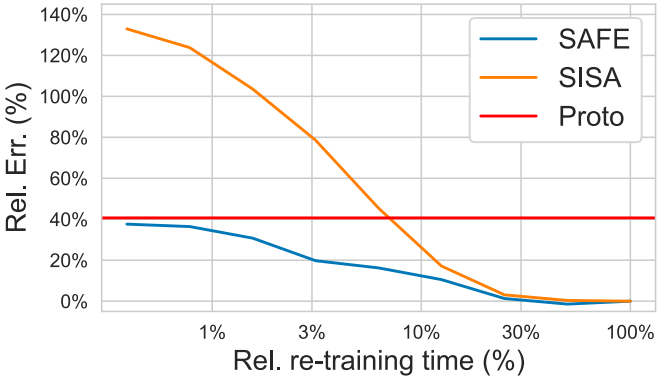


**Fig. 2.** Example of Pareto curves for different compartmentalization algorithms (reproduced from ref. 3). Based on the number of shards used to split the training data, compartmentalization algorithm implicitly makes a trade-off between the reduction in retraining time after a disgorgement request (x-axis) and the utility of the model (increase in test error relative to the paragon model, y-axis). Here, we report prototypical curves for three compartmentalization methods (3, 4, 24) averaging results over 7 different fine-grained visual classification tasks. Lower curves are better. Compartmentalization allows significant reduction in the cost to satisfy a forgetting request (up to only 0.3% of the baseline retraining from scratch). This, however, comes at the cost of increased error with respect to a reference model trained without compartmentalization (*Rightmost* point). Users can select the best trade-off for their applications, but better methods can improve the trade-off.

[‡]In general, often different samples of data sources often come with different safety requirements. Compartmentalization methods generally benefit from having a large core shard that is deemed safe, meaning that disgorgement requests are unlikely. Often, this is used to initial base network using the standard training method on top of whichunsafe data can be added through forgetting or compartmentalization techniques.

A further advantage of compartmentalization-based approaches is that disgorgement of data of an entire shard (or multiple shards)—as opposed to disgorgement of individual samples—can be performed at essentially zero cost. Hence, if one expects that all samples from a data source may need to be disgorged at the same time, it may be beneficial to group all data from that source into a single shard, which may then easily be dropped. This, however, may be difficult to realize if the source does not exhibit sufficient variety to enable training strong submodels. For example, if shards were organized by application domain, each model built on a homogeneous shard would overfit to that domain, resulting in a collection of biased models. Combining such models would likely lead to a severe performance degradation compared to an unconstrained model trained on a monolithic dataset.

Compartmentalization provides deterministic and structural guarantees, as it ensures by construction that the disgorged data will have zero influence on the disgorged model. However, it may be difficult to predict the cost of disgorgement and the reduction in utility after disgorgement, as they depend on a complex interaction between the model architecture, the sharding structure, and the distribution of samples to disgorge. Moreover, in the limit where the data to be disgorged touches all shards, this method requires retraining from scratch. In the next section, we discuss the alternative approach, which is to make the cost of disgorgement zero, by preempting the need to disgorge, in exchange for a probabilistic guarantee on the influence of the cohort of data in question.

## 3. Preemptive Disgorgement

Preemptive disgorgement refers to modifications of the training process that, by design, ensure that the influence of "unique information" (30) contained in any cohort of samples in the training data is bounded by a small value selected by the system designer. Since no substantial information about any training sample is present in the model, in principle nothing needs to be done to satisfy an individual disgorgement request. However, as we will discuss in Section 4, disgorgement of larger groups of data still presents a challenge to preemptive methods, as do high dimensional training data such as images whose variability can almost entirely be ascribed to nuisance factors. Based on this, we make recommendations to combine compartmentalization and differential privacy to address the most challenging cases. We also introduce dataset emulation, a new framework for preemptive disgorgement. We also use the method to highlight the challenges in defining and measuring the influence of a training sample in complex real-world data distributions.

**3.1. Differential Privacy.** In many ways, differential privacy (abbreviated DP in the sequel) (31) can be considered the "gold standard" of model disgorgement, in that it proactively trains a model in a way that provides a mathematical proof that no particular piece of training data had more than a negligible effect on the model or the output it generates. The technical details of the definition of DP are beyond the scope of this document, but its force is captured in the following thought experiment. Imagine the original training dataset $D$, and consider the dataset $D'$ that results from removing any small fraction of $D$ (say, all the works of a particular artist). Then under DP, a model trained on $D$ is (provably) statistically indistinguishable from one trained on $D'$ even to an observer who knows both $D$ and $D'$. In the generative setting, this means the distribution over output content for a given input prompt is also indistinguishable. In other words, a DP model effectively already disgorges any (small) amount of training data by minimizing its effects a priori.

DP models are achieved by deliberately adding noise to the training process (32–34) in a way that attempts to eradicate the impact of any small piece of the data while still having the desired aggregate effects (in generative models, high-quality outputs). In general, one expects that adding more noise provides stronger disgorgement and privacy properties but will also degrade output quality. This trade-off is determined by a privacy parameter that must be tuned and chosen.

The practice of DP model training remains in its infancy, and its effectiveness is untested on the scale of the hundreds of billions of parameters common in modern generative models. We should expect there to be challenges to the adoption of DP as a disgorgement solution, but it is a worthwhile standard for comparison for other approaches, and may eventually be a feasible solution. Moreover, DP can be combined with other methods, such as compartmentalization, and leverage a "safe set" to yield a more favorable privacy/performance trade-off (34).

**3.2. Dataset Emulation.** Differential privacy modifies the training algorithm to ensure little unique information about any individual training sample is contained in the final parameters. However, as mentioned, applying such training procedure to large models may still be challenging. Another factor to consider is that DP does not differentiate which unique information may be permissible to use. In particular, some disgorgement requests arise in the context of generative AI, where the concern is that the ML model may generate data that is "similar to" or "in the style of" data used for training. Practitioners may therefore want to prevent these characteristic elements of the training data to affect the model, but still use the remaining information to capture the general distribution of the data.

Dataset emulation aims to provide a different approach to preemptive disgorgement which allows practitioners more flexibility in deciding which information should be preemptively disgorged without requiring changes to the training pipeline. This is achieved by replacing the original training data $D$ with an "emulated" synthetic dataset $D_{\text{em}}$ which is constructed to capture the general distributional properties of the original training set $D$ while deliberately maximizing the geometric, perceptual, or conceptual distance from $D$, as defined below. Schematically, the dataset emulation pipeline may be represented as shown in Fig. 3.

While a synthetic dataset $D_{\text{em}}$ generated based on $D$ has a computational relationship with the latter, the generation pipeline can be designed to only capture general distributional properties while exercising maximum care to avoid


**Fig. 3.** Schematic workflow of dataset emulation.

generating samples close to $D$. Once such a guarantee is provided, the data $D_{\text{em}}$ may be usable in downstream tasks without using the protectable elements of $D$. In particular, the resulting data can be used to train a model, now without constraints, that can be used to discriminate or generate new outputs $D_{\text{gen}}$. In the latter case, the emulated dataset $D_{\text{em}}$ acts as a clean room or firewall to separate the downstream generative model generating $D_{\text{gen}}$ from the training dataset $D$, which not only has never been seen by the model that produces $D_{\text{gen}}$, but which has samples that are, by design, as different as possible from it. If so desired, users can manually inspect $D_{\text{em}}$ to ensure that transferred elements are not contained in it, or that it is sufficiently different, before the final model is trained (Fig. 4).

The complexity of preemptive disgorgement is now shifted to the generation of $D_{\text{em}}$. On the one hand, we want to ensure no sensitive information leaks to $D_{\text{em}}$, while on the other we want to ensure that $D_{\text{em}}$ is as similar as possible to $D$ in any other aspect, in order to reduce the sim-to-real gap (37, 38) and ensure that emulated data will be as useful as the original for the downstream tasks.

In some cases, the attributes of the training data $D$ may be relatively simple and it may be possible to hard-code an algorithm that removes that information from the samples of $D$ to generate new realistic synthetic data (e.g., replacing all phone numbers in a training sample with randomly generated ones). Another option is for $D_{\text{em}}$ to be generated in a different modality than $D$ and to consist of restricted abstract descriptions of the training images, for instance, textual descriptions or embedding vectors. In this case, the absence of protectable elements in the generated data is enforced structurally by ensuring that those elements cannot be captured by the restricted descriptions. However, such description may not be expressive enough and may not include all task-relevant information.

In other cases, however, hand-designed generators or descriptions are not possible and a more general learned model has to be trained to generate emulated samples that differ as much as possible from the original along the dimensions we want to protect. In these cases, we assume a distance function $d(x, x')$ is given to measure similarity among the chosen dimensions. Such a distance can be used either at training time to force the model to generate sufficiently different data, or at inference time to reject samples that are too close to the original. As shown in Fig. 4 this method is effective at generating emulated datasets for human evaluation, but is not yet viable as a substitute for the original dataset to train a generative or discriminative model due to the above-mentioned sim-to-real gap. However, we believe that steady progress in the area of unsupervised

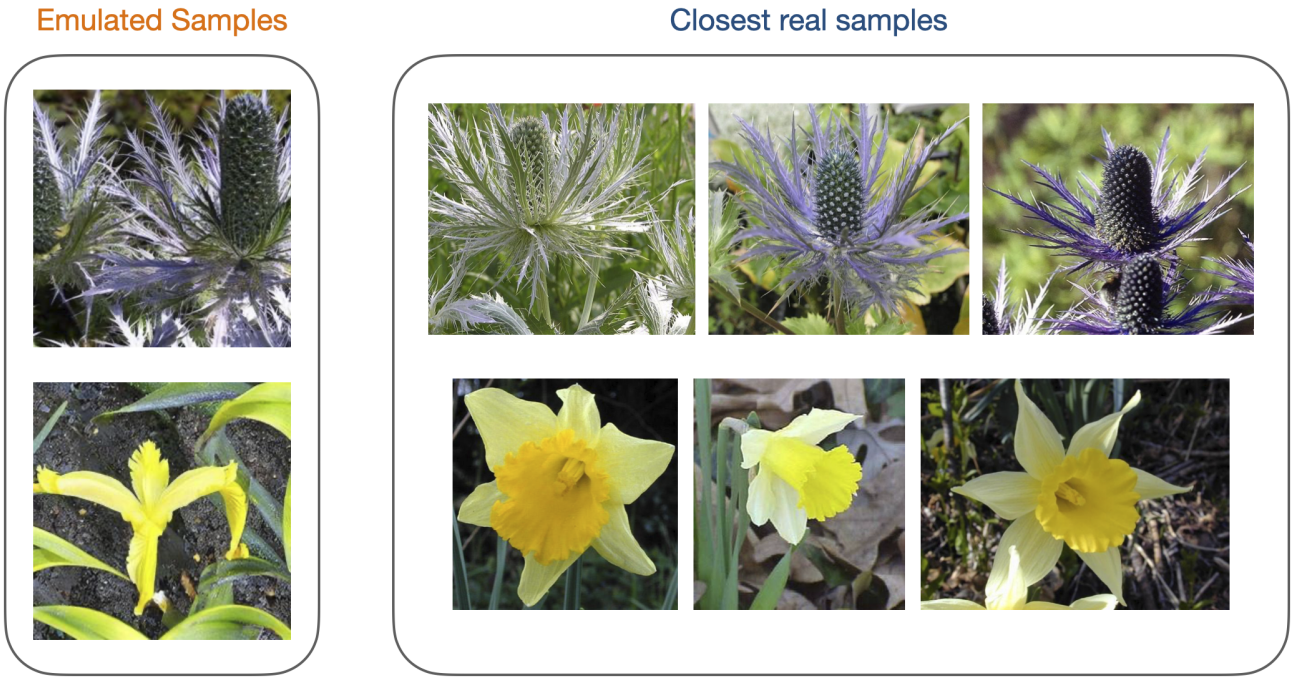


**Fig. 4.** Example of dataset emulation. We show samples of an emulated dataset of Oxford Flowers (35), which captures the original distribution while maintaining high CLIP distance (36) from the original data. We note that $D_{\text{em}}$ captures the concepts but is composed of substantially novel images that do not imitate the original training data. However, while $D_{\text{em}}$ has good visual quality, due to the sim-to-real gap training a model on it for a downstream flower classification task leads to a low 52% classification accuracy, catastrophically lower than what achieved training directly on $D$ (96%).

domain adaptation may make the technique viable in the near future.

Defining a distance that is able to encode perceived similarity between samples is also challenging. For symbolic data such as text (represented either in raw form or as a vector embedding $x$), there are objective ("geometric") distances $d_g(x_1, x_2) = \|x_2 - x_1\|$ that approximate perceived similarity well. However, for signal data such as images, any datum $x_1$ has a large number of samples that are far from $x_1$ according to $d_g$, yet are perceptually indistinguishable. This can occur both for subtle changes (39) or for macroscopic ones (40). Indeed, one can significantly change the value of every pixel in an image without triggering a perceptible change. To capture perceptual, rather than geometric, similarity, one can compare low-level statistics designed to mimic the early stages of human cortical processing, designed so that data points that have small distance are effectively (with high probability) indistinguishable by humans (41). One could also imagine training distance metrics capturing human perceptions about content similarity (e.g., text passages viewed as "in the style" of a particular author).

The choice of distance $d(x, x')$ and a threshold $d(x, x') \leq r$ is critical as it implicitly defines what outputs $x'$ should be considered a "reproduction" of the data $x$. For example, say a vendor who owns $D$ is interested in licensing it for training to a service provider who already has a model trained on the dataset $D_{\text{em}}$. The vendor may require that all data generated be farther than some $r > 0$ from $D$ according to a chosen distance $d$. Clearly, if the images already generated by the service fall within a distance $r$, despite having been obtained without any knowledge of $D$, then the owner of $D$ would be attempting to control the usage of data that does not belong to them. Hence, the distance and threshold require careful calibration and testing to avoid under- or overreach. Even so, a distance could be fooled into misclassifying data as being sufficiently distant by applying imperceptible perturbations (42), although distances can be devised that are robust to such perturbations (43, 44).

## 4. Discussion

The methods we have surveyed vary in their applicability and value, depending on the intended application and the need to disgorge data. Below we discuss some important considerations:

**4.1. Preemptive Differential Privacy and Group Forgetting.** Consider the case where a group of related data points needs to be disgorged from a model. While DP ensures that the influence about any individual sample is negligible, the bound on the influence can quickly become vacuous as the size of the group increases. However, how this should be interpreted depends on the composition of the group. Depending on the size of the group and the initially selected privacy level, it may not be possible to guarantee negligible influence via DP as measured at the data group level. Notions such as user-level DP aim to address this problem, by grouping data of the same entity during training. However they require a priori knowledge during training of which samples are grouped, and what the composition of those groups is. For large-scale data usage, being able to group images along each possible grouping dimension may not be realistic, and no single grouping may be correct. This suggests using preemptive methods together with proactive methods (see next paragraph): The first minimizes the number of retrainings, the latter allows retraining on only smaller portions of the models.

**4.2. Compartmentalization and Group Forgetting.** Compartmentalization methods are particularly well-suited to scenarios where a significant portion of data needs to be disgorged together. For example, the terms of a limited data license may require an entire dataset to be disgorged after a period of time, where each of the many data points in the cohort has some shared information, for instance, a watermark. If the entire cohort was compartmentalized in a single shard, then disgorgement is trivially accomplished by removing that shard and every submodel that used it during the training process. Provided that the model was originally trained with a sufficiently large "safe core" set of data, the impact on performance should remain limited. The key issue in disgorgement is how to shard the data. There is a vast design space that affects the potential cost of disgorgement and impact on the model performance. In addition to performance, model bias is also a concern, especially if shards are segregated by class or domain, for the resulting models would overfit, and simplistic ensembling of submodel activations is not likely to be sufficient to correct such biases. For this reason, shards should be sufficiently diverse to ensure the quality of submodels trained on them. We note however that, while better sharding may decrease the cost of forgetting and improve model performance, compartmentalization provides guaranteed deterministic forgetting regardless of the sharding used.

**4.3. Interpretation of Negligible Influence.** Both probabilistic unlearning and preemptive DP rely on bounds on the remaining influence of samples. For example, $(\epsilon, \delta)$-DP bounds the influence of individual samples through two tunable parameters $\epsilon$ and $\delta$ (the discussion for probabilistic unlearning and forgetting is similar). Smaller $(\epsilon, \delta)$ correspond to less influence, but it may not always be straightforward to translate this into user-facing measures. One strong guarantee is that, if $D$ and $D'$ are two datasets differing by a single example, an observer who knows both cannot design a statistical test at significance level $\alpha$ with power greater than $e^{\epsilon}\alpha + \delta$ (i.e., as mentioned above, one cannot determine with confidence if a sample was used to train the model or not) (45). Another important user-facing metric is how much the output of a model trained with or without a sample would differ. Smaller $(\epsilon, \delta)$ ensure that a model trained with a sample will produce a similar output to a model trained without the sample. However, exact quantification of similarity is difficult since human conceptual perception of similarity (especially in images) does not relate directly to metric similarity (e.g., difference in pixel values or log-likelihood of the output). In particular, exceedingly small values of $\epsilon$ may be needed to ensure that the outputs are conceptually similar.

**4.4. Interpretation of Forgetting and Model Outputs.** Yet another consideration is the mismatch between what it means to forget a sample and what the user may expect from

the resulting model. A disgorged model may still produce outputs that are similar (or conceptually similar) to the disgorged data. This is not a bug: for example, a string of text may be the likely answer to a prompt regardless of whether it was observed during training. One could consider, however, being more proactive and avoiding outputs too similar to disgorged data (even if this would, paradoxically, make the model less private with respect to that data). Dataset emulation follows a similar approach.

**4.5. Preemptive Methods, Model Performance, Long Tails, and Fairness.** By design, preemptive methods reduce influence of any samples. This has some unavoidable consequences on the performance of the model on long-tailed and undersampled data. On large-scale data, many (or most) tasks and domains may be represented by few (important) datapoints. During normal training, those datapoints are highly influential for the behavior of the model on their subpopulation. However, this is not acceptable for DP training, leading to these datapoints being effectively discounted. This may reduce accuracy of the model on long-tail tasks (a main selling point of foundational models) and on underrepresented populations (thus creating a fairness risk). In fact, some have advanced the notion that memorization of the long-tails (the opposite of DP) is essential for good performance of ML models (46), in some cases provably so (47).

**4.6. Inference with Compartmentalized Models.** Aside from facilitating disgorgement, compartmentalization also enables inference time protection. For example, ref. 48 trains a collection of generative models on sharded data. A sample is then generated by sampling from the "intersection" of the output probability of each model. Since no model has seen all training data, under their modeling assumptions this ensures that no training sample can have excessive influence on the generated output, thus reducing the risk of reproducing training data. We note however that while this realizes a form of differential privacy for the model's outputs, information about the training samples will still be contained in the weights of the model.

Ultimately, Model Disgorgement is only a small portion of a comprehensive process of ensuring that AI Models are trained responsibly. Careful data collection, curation, documentation, and provenance verification are always the starting points. Model Disgorgement is not intended as a substitute for those, but rather an additional option if the need or desire to eliminate a cohort of data and their effects becomes manifest.

**Data, Materials, and Software Availability.** There are no data underlying this work.

**ACKNOWLEDGMENTS.** The authors warmly acknowledge helpful conversations with Aaron Roth.

1. W. article, Disgorgement (2024). https://en.wikipedia.org/wiki/Disgorgement.
2. J. A. Goland, Algorithmic disgorgement: Destruction of artificial intelligence models as the FTC's newest enforcement tool for bad data. *Richm. J. Law Technol.* **XXIX** (2023).
3. Y. Dukler *et al*., Safe: Machine unlearning with shard graphs. arXiv [Preprint] (2023). https://arxiv.org/abs/2304.13169 (Accessed 31 December 2023).
4. L. Bourtoule *et al*., "Machine unlearning" in *2021 IEEE Symposium on Security and Privacy (SP)* (2021), pp. 141–159.
5. Y. Wu, E. Dobriban, S. Davidson, "Deltagrad: Rapid retraining of machine learning models" in *International Conference on Machine Learning* (PMLR, 2020), pp. 10355–10366.
6. Y. Shen, Y. Xiong, W. Xia, S. Soatto, "Towards backward-compatible representation learning" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 6368–6377.
7. M. Srivastava, B. Nushi, E. Kamar, S. Shah, E. Horvitz, "An empirical analysis of backward compatibility in machine learning systems" in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), pp. 3272–3280.
8. G. Bansal *et al*., "Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff" in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2019), vol. 33, pp. 2429–2437.
9. S. Yan *et al*., "Positive-congruent training: Towards regression-free model updates" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 14299–14308.
10. A. Golatkar, A. Achille, S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).
11. A. Golatkar, A. Achille, A. Ravichandran, M. Polito, S. Soatto, "Mixed-privacy forgetting in deep networks" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 792–801.
12. A. Golatkar, A. Achille, S. Soatto, "Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations" in *European Conference on Computer Vision* (Springer, 2020), pp. 383–398.
13. Y. Cao, J. Yang, "Towards making systems forget with machine unlearning" in *2015 IEEE Symposium on Security and Privacy* (2015), pp. 463–480.
14. A. Ginart, M. Guan, G. Valiant, J. Y. Zou, Making ai forget you: Data deletion in machine learning. *Adv. Neural Inf. Process. Syst.* **32** (2019).
15. T. B. Shaik *et al*., Exploring the landscape of machine unlearning: A comprehensive survey and taxonomy. arXiv [Preprint] (2023). https://arxiv.org/abs/2305.06360 (Accessed 31 December 2023).
16. S. Neel, A. Roth, S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning" in *Algorithmic Learning Theory* (PMLR, 2021), pp. 931–962.
17. V. Gupta *et al*., Adaptive machine unlearning. *Adv. Neural Inf. Process. Syst.* **34**, 16319–16330 (2021).
18. C. Guo, T. Goldstein, A. Hannun, L. Van Der Maaten, Certified data removal from machine learning models. arXiv [Preprint] (2019). http://arxiv.org/abs/1911.03030 (Accessed 31 December 2023).
19. P. W. Koh, P. Liang, "Understanding black-box predictions via influence functions" in *International Conference on Machine Learning* (PMLR, 2017), pp. 1885–1894.
20. S. Basu, P. Pope, S. Feizi, Influence functions in deep learning are fragile. arXiv [Preprint] (2020). http://arxiv.org/abs/2006.14651 (Accessed 31 December 2023).
21. M. Pawelczyk, S. Neel, H. Lakkaraju, In-context unlearning: Language models as few shot unlearners. arXiv [Preprint] (2023). http://arxiv.org/abs/2310.07579 (Accessed 31 December 2023).
22. J. Jang *et al*., Knowledge unlearning for mitigating privacy risks in language models. arXiv [Preprint] (2022). http://arxiv.org/abs/2210.01504 (Accessed 31 December 2023).
23. D. Yoon, J. Jang, S. Kim, M. Seo, Gradient ascent post-training enhances language model generalization. arXiv [Preprint] (2023). http://arxiv.org/abs/2306.07052 (Accessed 31 December 2023).
24. H. Yan *et al*., "Arcane: An efficient architecture for exact machine unlearning" in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, L. D. Raedt, Ed. (2022), pp. 4006–4013. Main Track.
25. X. Zhu *et al*., Exploring user historical semantic and sentiment preference for microblog sentiment classification. *Neurocomputing* **464**, 141–150 (2021).
26. K. Koch , M. Soll , "No matter how you slice it: Machine unlearning with sisa comes at the expense of minority classes" in *First IEEE Conference on Secure and Trustworthy Machine Learning* (2022).
27. V. B. Kumar, R. Gangadharaiah, D. Roth, Privacy adhering machine un-learning in NLP. arXiv [Preprint] (2022). http://arxiv.org/abs/2212.09573 (Accessed 31 December 2023).
28. N. Vyas, S. Kakade, B. Barak, Provable copyright protection for generative models. arXiv [Preprint] (2023). http://arxiv.org/abs/2302.10870 (Accessed 31 December 2023).
29. Y. Dukler *et al*., Introspective cross-attention probing for lightweight transfer of pre-trained models. arXiv [Preprint] (2023). http://arxiv.org/abs/2303.04105 (Accessed 31 December 2023).
30. H. Harutyunyan *et al*., Estimating informativeness of samples with smooth unique information. arXiv [Preprint] (2021). http://arxiv.org/abs/2101.06640 (Accessed 31 December 2023).
31. C. Dwork, A. Roth, The algorithmic foundations of differential privacy. *Found. Trend. Theor. Comput. Sci.* **9**, 211–407 (2014).
32. M. Abadi *et al*., "Deep learning with differential privacy" in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), pp. 308–318.
33. Z. Bu, Y. X. Wang, S. Zha, G. Karypis, Differentially private optimization on large model at small cost. arXiv [Preprint] (2022). http://arxiv.org/abs/2210.00038 (Accessed 31 December 2023).
34. A. Golatkar *et al*., "Mixed differential privacy in computer vision" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 8376–8386.
35. M. E. Nilsback, A. Zisserman, "A visual vocabulary for flower classification" in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)* (IEEE, 2006), vol. 2, pp. 1447–1454.
36. A. Radford *et al*., "Learning transferable visual models from natural language supervision" in *International Conference on Machine Learning* (PMLR, 2021), pp. 8748–8763.
37. W. Zhao, J. P. Queralta, T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey" in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, 2020), pp. 737–744.
38. Q. Xie, Z. Dai, E. Hovy, T. Luong, Q. Le, Unsupervised data augmentation for consistency training. *Adv. Neural Inf. Process. Syst.* **33**, 6256–6268 (2020).
39. Y. Wu, W. Ji, J. Wu, "Unsupervised deep learning for just noticeable difference estimation" in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (2020), pp. 1–6.
40. R. A. Rensink, "Change blindness: Implications for the nature of visual attention" in *Vision and Attention* (2001), pp. 169–188.
41. Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **13**, 600–612 (2004).

42. S. M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, "Universal adversarial perturbations" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1765–1773.
43. Y. Zeng *et al*., "Towards robustness certification against universal perturbations" in *The Eleventh International Conference on Learning Representations* (2023).
44. S. Shan *et al*., Glaze: Protecting artists from style mimicry by text-to-image models. arXiv [Preprint] (2023). http://arxiv.org/abs/2302.04222 (Accessed 31 December 2023).
45. J. Dong, A. Roth, W. J. Su, Gaussian differential privacy. *J. R. Stat. Soc. Ser. B: Stat. Methodol.* **84**, 3–37 (2022).
46. V. Feldman, "Does learning require memorization? A short tale about a long tail" in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (2020), pp. 954–959.
47. G. Brown, M. Bun, V. Feldman, A. Smith, K. Talwar, "When is memorization of irrelevant training data necessary for high-accuracy learning?" in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (2021), pp. 123–132.
48. N. Vyas, S. Kakade, B. Barak, "Provable copyright protection for generative models" in *Proceedings International Conference for Machine Learning* (2023).