Forgetting Data from Pre-trained GANs

Anonymous Author(s)
Affiliation
Address
Email

***Abstract*—Large pre-trained generative models are known to occasionally output undesirable samples, which contributes to their trustworthiness. The common way to mitigate this is to re-train them differently from scratch using different data or different regularization – which uses a lot of computational resources and does not always fully address the problem.**

In this work, we take a different, more compute-friendly approach and investigate how to post-edit a model after training so that it "forgets", or refrains from outputting certain kinds of samples. We show that forgetting is different from data deletion, and data deletion may not always lead to forgetting. We then consider Generative Adversarial Networks (GANs), and provide three different algorithms for data forgetting that differ on how the samples to be forgotten are described. Extensive evaluations on real-world image datasets show that our algorithms out-perform data deletion baselines, and are capable of forgetting data while retaining high generation quality at a fraction of the cost of full re-training.

***Index Terms*—Data forgetting, post-editing, pre-trained GANs**

I. Introduction

Generative Adversarial Networks (GANs) are large neural generative models that learn a complicated probability distribution from data and then generate samples from it. These models have been immensely successful in many large scale tasks from multiple domains, such as images [Zhu et al., 2020, Karras et al., 2020, 2021], point clouds [Zhang et al., 2021], video [Tulyakov et al., 2018], text [de Masson d'Autume et al., 2019], and speech [Kong et al., 2020].

However, it is also well-known that many deep generative models frequently output undesirable samples, which makes them less reliable and trustworthy. Image models generate blurred samples [Kaneko and Harada, 2021] or checkerboard artifacts [Odena et al., 2016, Zhang et al., 2019, Wang et al., 2020, Schwarz et al., 2021], speech models produce unnatural sound [Donahue et al., 2018, Thiem et al., 2020], and language models emit offensive text [Abid et al., 2021, Perez et al., 2022]. Thus, an important question is how to mitigate these artifacts, which would improve the trustworthiness of these models.

One way to mitigate undesirable samples is to re-design the entire training pipeline including data augmentation, model architecture and loss functions, and then re-train the entire model from scratch [Isola et al., 2017, Aitken et al., 2017, Kaneko and Harada, 2021] – a strategy that has been used in prior work. This approach is very compute-intensive as modern GANs can be extremely expensive to train. In addition, other problems may become apparent after training, and resolving them may require multiple re-trainings. To address this challenge, we consider *post-editing*, which means modifying a pre-trained model in a certain way rather than training it differently from scratch. This is a much more computationally efficient process that has shown empirical success in many supervised learning tasks [Frankle and Carbin, 2018, Zhou et al., 2021, Taha et al., 2021], but has not been studied much for unsupervised learning. In particular, we propose a post-editing framework to forget undesirable samples that might be generated by a GAN, which we call *data forgetting*.

A second plausible solution for mitigating undesirable samples is to use a classifier to filter them out after generation. This approach, however, has several drawbacks. Classifiers can take a significant amount of space and time after deploymen. Additionally, if the generative model is handed to a third party,

then the model trainer has no control over whether the filter will ultimately be used. Data forgetting via post-editing, on the other hand, offers a cleaner solution which does not suffer from these limitations.

A third plausible solution is data deletion or machine unlearning – post-edit the model to approximate a re-trained model that is obtained by re-training from scratch after removing the undesirable samples from the training data. However, this does not always work – as we show in Section IV-B, deletion does not necessarily lead to forgetting in constrained models. Additionally, the undesirable samples may simply be artifacts of the neural generative model and may not exist in the training data; examples include unnatural sounds emitted by speech models and blurred images from image models. Data forgetting, in contrast, can address all these challenges.

There are two major technical challenges that we need to resolve in order to do effective data forgetting. The first is how to describe the samples to be forgotten. This is important as data forgetting algorithms need to be tailored to specific descriptions. The second challenge is that we need to carefully balance data forgetting with retaining good generation quality, which means the latent space and the networks must be carefully manipulated.

In this work, we propose a systematic framework for forgetting data from pre-trained generative models (see Section II). We model data forgetting as learning the data distribution restricted to the complement of a forgetting set Ω. We then formalize three ways of describing forgetting sets, namely data-based (where a pre-specified set is given), validity-based (where there is a validity checker), and classifier-based (where there is a differentiable classifier).

Then, we introduce three data forgetting algorithms, one for each description (see Section III). Prior works have looked at avoiding negative samples in the re-training setting with different descriptions and purposes [Sinha et al., 2020, Asokan and Seelamantula, 2020]. They introduce fake distributions to penalize the generation of negative samples. We extend this idea to data forgetting by defining the fake distribution as a mixture of the generative distribution and a forgetting distribution supported on Ω. We prove the optimal generator can recover the target distribution when label smoothing [Salimans et al., 2016, Szegedy et al., 2016, Warde-Farley and Goodfellow, 2016] is used.

Based on our theory, we introduce the data-based forgetting algorithm (Alg. 1). We then combine this algorithm with an improper active learning algorithm by Hanneke et al. [2018] and introduce the validity-based forgetting algorithm (Alg. 2). Finally, we propose to use a `guide` function to guide the discriminator via a classifier, and introduce the classifier-based forgetting algorithm (Alg. 3).

Finally, we empirically evaluate these forgetting algorithms via experiments on real-world image datasets (see Section IV). We show that these algorithms can forget quickly while keeping high generation quality. We then investigate applications of data forgetting, and use our algorithms to remove different biases that may not exist in the training set but are learned by the pre-trained model. This demonstrates that data forgetting can be used to reduce biases and improve generation quality, and hence improve the trustworthiness of generative models.

In summary, our contributions are as follows:

- We formalize the problem of post-editing generative models to prevent them from outputting undesirable samples as "data forgetting" and establish its differences with data deletion.
- We propose three data augmentation-based algorithms for forgetting data from pre-trained GANs as a function of how the inputs to be forgotten are described.
- We theoretically prove that data forgetting can be achieved by the proposed algorithms.
- We extensively evaluate our algorithms on real world image datasets. We show these algorithms can forget data quickly while retaining high generation quality. Moreover, we find data forgetting performs better than data deletion in a de-biasing experiment.

II. A Formal Framework for Data Forgetting

Let p_{data} be the data distribution on $\mathbb{R}^d$ and $X \sim p_{\text{data}}$ be i.i.d. training samples. Let $\mathcal{A}$ be a generative model and $\mathcal{M} = \mathcal{A}(X)$ be the pre-trained model on X, which learns p_{data}. In this paper, we consider $\mathcal{A}$ to be a GAN learning algorithm [Goodfellow et al., 2014a], and $\mathcal{M}$ contains two networks, D (discriminator) and G (generator), which are jointly trained to optimize

$$\min_G \max_D \ \mathbb{E}_{x\sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z\sim\mathcal{N}(0,I)} \log(1 - D(G(z))). \quad (1)$$

A. Data Forgetting Framework

Let the forgetting set $\Omega \subset \mathbb{R}^d$ be the set of samples we would like the model to forget. Formally, the goal is to develop a forgetting algorithm $\mathcal{D}$ such that $\mathcal{M}' = \mathcal{D}(\mathcal{M}, \Omega)$ learns the data distribution restricted to the complement $\bar{\Omega} = \mathbb{R}^d \setminus \Omega$, i.e. $p_{\text{data}}|_{\bar{\Omega}}$. Examples of Ω include inconsistent, blurred, unrealistic, or banned samples that are possibly generated by the model.

The forgetting set Ω, in addition to the pre-trained model, is considered as an input to the forgetting algorithm. We consider three kinds of Ω, namely data-based, validity-based, and classifier-based.

B. Forgetting Set Descriptions

We propose three different descriptions for the forgetting set Ω. First, the data-based Ω is a pre-defined set of samples in $\mathbb{R}^d$, such as a transformation applied on all training samples [Sinha et al., 2020]. Second, the validity-based Ω is defined as all invalid samples according to a validity function $\mathbf{v} : \mathbb{R}^d \to \{0, 1\}$, where 0 means invalid and 1 means valid. This is similar to the setting in Hanneke et al. [2018]. Finally, let $\mathbf{f} : \mathbb{R}^d \to [0, 1]$ be a soft classifier that outputs the probability that a sample belongs to a certain binary class, and $\tau \in (0, 1)$ be a threshold. Then, the classifier-based Ω is defined as $\{x : \mathbf{f}(x) < \tau\}$. For example, $\mathbf{f}$ can be an offensive text classifier in language generation tasks [Pitsilis et al., 2018]. Our framework is general and applies to any kind of generative model besides GANs.

C. Data Deletion versus Data Forgetting

Motivated by privacy laws such as the GDPR and the CCPA, there has been a recent body of work on data deletion or machine unlearning [Cao and Yang, 2015, Guo et al., 2019, Schelter, 2020, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021, Ullah et al., 2021]. In data deletion, we are given a subset set $X' \subset X$ of the training set to be deleted from an already-trained model, and the goal is to approximate the re-trained model $\mathcal{A}(X \setminus X')$. While there are some superficial similarities – in that the goal is to post-edit models in order to "remove" a few data points, there are two very important differences.

The first is that data forgetting requires the model to assign zero likelihood to the forgetting set Ω in order to avoid generating samples from this region; this is not the case in data deletion – in fact, we present an example below which shows that data deletion of a set X' may not cause a generative model to forget X'.

Specifically, in Fig. 1, the entire data distribution $p_{\text{data}} = \mathcal{N}(0, 1)$ (blue line) is the standard Gaussian distribution on $\mathbb{R}$. We set the forgetting set $\Omega = (-\infty, -1.5] \cup [1.5, \infty)$, so the blue samples fall in Ω and orange samples outside. The learning algorithm $\mathcal{A}$ is the maximum likelihood Gaussian learner that fits the mean and variance of the data. With $n = 80$ samples, the learnt density $\mathcal{A}(X)$ is shown in green. If the blue samples were **deleted**, and the model re-fitted, the newly learnt density $\mathcal{A}(X \setminus X')$ would be the red line. Notice that this red line has considerable density on the blue points – and so these points are not forgotten. In contrast, the correct **forgetting** solution that forgets the samples in Ω would be the orange density. Thus deletion does not necessarily lead to forgetting.

The second difference is that the forgetting set Ω may have a *zero intersection* with the training data, but may appear in the generated data due to artifacts of the model. Examples include unnatural sounds emitted by speech models, and blurred images from image models. Data forgetting, in contrast to data deletion, can address this challenge.

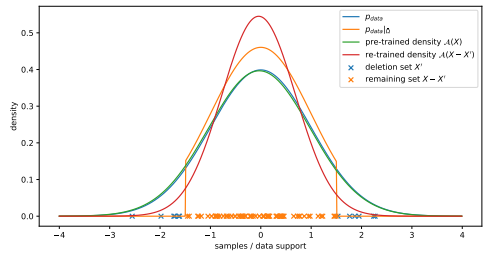


Fig. 1: An example showing difference between data forgetting and data deletion. The goal of **data deletion** is to approximate the re-trained model (red density), while the goal of **data forgetting** is to approximate the restricted density (orange density).

III. Methods

In this section, we describe algorithms for each kind of forgetting set described in Section II. We also provide theory on the optimality of the generator and the discriminator. Finally, we generalize the algorithms to situations where we would like the model to forget the union of multiple forgetting sets.

A. Data-based Forgetting Set

The data-based forgetting set Ω is a pre-defined set of samples we would like the model to forget. One example is a transformation function `NegAug` applied to all training samples, where `NegAug` makes realistic images unrealistic or inconsistent [Sinha et al., 2020]. Another example can be visually nice samples outside data manifold when the training set is small [Asokan and Seelamantula, 2020].

In our framework, the forgetting set Ω can be any set of carefully designed or selected samples depending on the purpose of forgetting them – which includes but does not limit to improving the generation quality of the model. For example, we expect the model to improve on fairness, bias, ethics or privacy when Ω is properly constructed with unfair, biased, unethical, or atypical samples.

To forget Ω, we regard both generated samples and Ω to be fake samples, and all training samples that are not in Ω to be real samples [Sinha et al., 2020, Asokan and Seelamantula, 2020]. Let p_Ω be a distribution such that $\mathbf{supp}(p_\Omega) = \Omega$. Then, the fake data distribution p_{fake} is a mixture of the generative distribution $p_G = G\#\mathcal{N}(0, I)$ and p_Ω:

$$p_{\text{fake}} = \lambda \cdot p_G + (1 - \lambda) \cdot p_\Omega, \tag{2}$$

where $\lambda \in (0, 1)$ is a hyperparameter. We also apply label smoothing [Salimans et al., 2016, Szegedy et al., 2016, Warde-Farley and Goodfellow, 2016] techniques to the minimax loss function in order to improve robustness. Let $\alpha_+ \in (\frac{1}{2}, 1]$ be the positive target (such as 0.9) and $\alpha_- \in [0, \frac{1}{2})$ be the negative target (such as 0.1). Then, the loss function is

$$\begin{aligned} L(G, D) = \; & \mathbb{E}_{x \sim p_{\text{data}}|_{\bar{\Omega}}} \left[\alpha_+ \log D(x) \right. \\ & \left. + (1 - \alpha_+) \log(1 - D(x))\right] \\ + \; & \mathbb{E}_{x \sim p_{\text{fake}}} \left[\alpha_- \log D(x) \right. \\ & \left. + (1 - \alpha_-) \log(1 - D(x))\right]. \end{aligned} \tag{3}$$

Theorem 1. *The optimal solution to* $\min_G \max_D L(G, D)$ *is*

$$\begin{aligned} D^* &= \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1 - \lambda) p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1 - \lambda) p_\Omega} \\ p_{G^*} &= p_{\text{data}}|_{\bar{\Omega}} \end{aligned}. \tag{4}$$

We provide the proof and theoretical extension to the more general f-GAN [Nowozin et al., 2016] setting in Appendix A. In the data-based setting, we let $p_\Omega = \mathcal{U}(\Omega)$, the uniform distribution on Ω. We assume Ω has positive, finite Lebesgue measure in $\mathbb{R}^d$ so that $\mathcal{U}(\Omega)$ is well-defined. The proposed method is summarized in Alg. 1.

Our objective function is connected to Sinha et al. [2020] and Asokan and Seelamantula [2020] in the sense that p_Ω is an instance of the negative distribution described in their frameworks. However, there are several significant differences between our method and theirs: (1) we start from a pre-trained model, (2) we aim to learn $p_{\text{data}}|_{\bar{\Omega}}$ rather than p_{data} and therefore do not require $\Omega \cap \mathbf{supp}(p_{\text{data}})$ to be the empty set, and (3) we use label smoothing techniques to improve robustness and provide theory for this setting. These differences are also true in the following sections.

B. Validity-based Forgetting Set

Let $\mathbf{v} : \mathbb{R}^d \to \{0, 1\}$ be a validity function that indicates whether a sample is valid. Then, validity-

Algorithm 1 Forgetting Algorithm for Data-based Forgetting Set

Inputs: Pre-trained model $\mathcal{M} = (G_0, D_0)$, train set X, forgetting set Ω.
Initialize $G = G_0$, $D = D_0$.
Define the fake data distribution p_{fake} according to (2) with $p_\Omega = \mathcal{U}(\Omega)$.
Train G, D to optimize (3): $\min_G \max_D L(G, D)$.
return $\mathcal{M}' = (G, D)$.

Algorithm 2 Forgetting Algorithm for Validity-based Forgetting Set

Inputs: Pre-trained model $\mathcal{M} = (G_0, D_0)$, train set X, validity function $\mathbf{v}$.
Initialize $\Omega' = \{x \in X : \mathbf{v}(x) = 0\}$, $\mathcal{M}_0 = \mathcal{M}$.
for $i = 0, \cdots, R-1$ **do**
  Initialize $G = G_i$, $D = D_i$. Draw T samples $X_{\text{gen}}^{(i)}$ from G_i.
  Query $\mathbf{v}$ and add invalid samples to Ω': $\Omega' \leftarrow \Omega' \cup \{x \in X_{\text{gen}}^{(i)} : \mathbf{v}(x) = 0\}$.
  Define the fake data distribution p_{fake} according to (2) with $p_\Omega = \mathcal{U}(\Omega')$.
  Let $\mathcal{M}_{i+1} = (G_{i+1}, D_{i+1})$ optimize (3): $\min_G \max_D L(G, D)$.
end for
return $\mathcal{M}' = (G_R, D_R)$

Algorithm 3 Forgetting Algorithm for Classifier-based Forgetting Set

Inputs: Pre-trained model $\mathcal{M} = (G_0, D_0)$, train set X, differentiable classifier $\mathbf{f}$.
Initialize $G = G_0$, $D = D_0$.
Define the fake data distribution p_{fake} according to (2) with $p_\Omega = \mathcal{U}(\{x \in X : \mathbf{f}(x) < \tau\})$.
Train G, D to optimize (3): $\min_G \max_D L(G, \texttt{guide}(D, \mathbf{f}))$, where $\texttt{guide}(\cdot, \cdot)$ is defined in (6).
return $\mathcal{M}' = (G, D)$.

based forgetting set Ω is the set of all invalid samples $\{x : \mathbf{v}(x) = 0\}$. For example, $\mathcal{M}$ is a code generation model, and $\mathbf{v}$ is a compiler that indicates whether the code is free of syntax errors [Hanneke et al., 2018]. Different from the data-based setting, the validity-based Ω may have infinite Lebesgue measure, such as a halfspace, and consequently $\mathcal{U}(\Omega)$ may not be well-defined.

To forget Ω, we let p_Ω in (2) to be a mixture of $p_{\text{data}}|_\Omega$ and $p_G|_\Omega$. This corresponds to a simplified version of the improper active learning algorithm introduced by Hanneke et al. [2018] with our Alg. 1 as their optimization oracle. The idea is to apply Alg. 1 for R rounds. After each round, we query the validity of T newly generated samples and use invalid samples to form a data-based forgetting set Ω'. In contrast to the data-based approach, this active algorithm focuses on invalid samples that are more likely to be generated, and therefore efficiently penalizes generation of invalid samples. The proposed method is summarized in Alg. 2.

The total number of queries to the validity function $\mathbf{v}$ is $|X| + T \times R$. In case $\mathbf{v}$ is expensive to run, we would like to achieve better data forgetting within a limited number of queries. From the data-driven point of view, we hope to collect as many invalid samples as possible. This is done by setting $R = 1$ and T maximized if we assume less invalid samples are generated after each iteration. However, this may not be the case in practice. We hypothesis some samples are easier to forget while others harder. By setting $R > 1$, we expect an increasing fraction of invalid generated samples to be hard to forget after each iteration. Focusing on these hard samples can potentially help the generator forget them. Since it is hard to directly analyze neural

networks, we leave the rigorous study to future work. In Appendix B, we study a much simplified dynamical system corresponding to Alg. 2, where we show the invalidity (the mass of p_G on Ω) converges to zero, and provide optimal T and R values.

C. Classifier-based Forgetting Set

We would like the model to forget samples with certain (potentially undesirable) property. Let $\mathbf{f} : \mathbb{R}^d \to [0, 1]$ be a soft binary classifier on the property (0 means having the property and 1 means not having it), and $\tau \in (0, 1)$ be a threshold. The classifier-based forgetting set Ω is then defined as $\{x : \mathbf{f}(x) < \tau\}$. For example, the property can be *being offensive* in language generation, *containing no speech* in speech synthesis, or *visual inconsistency* in image generation. We consider $\mathbf{f}$ to be a trained machine learning model that is fully accessible and differentiable.

To forget Ω, we let p_Ω be a mixture of $p_{\text{data}}|_\Omega$ and $p_G|_\Omega$, similar to the validity-based approach. We use $\mathbf{f}$ to guide the discriminator and make it able to easily detect samples from Ω. Let $\texttt{guide}(D, \mathbf{f})$ be a guided discriminator that assigns small values to x when $\mathbf{f}(x) < \tau$ or $D(x)$ is small (i.e. $x \sim p_{\text{fake}}$), and large values to x when $\mathbf{f}(x) > \tau$ and $D(x)$ is large (i.e. $x \sim p_{\text{data}}|_{\bar{\Omega}}$). Instead of optimizing $L(G, D)$ in (3), we optimize $L(G, \texttt{guide}(D, \mathbf{f}))$. This will effectively update G by preventing it from generating samples in Ω. According to **Theorem** 1, the optimal discriminator is the solution to

$$\texttt{guide}(D^*, \mathbf{f}) = \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1-\lambda) p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1-\lambda) p_\Omega}. \quad (5)$$

Therefore, the design of the `guide` function must make (5) feasible. In this paper, we let

$$\texttt{guide}(D, \mathbf{f})(x) = \begin{cases} D(x) & \text{if } \mathbf{f}(x) \geq \tau \\ \alpha_- + (D(x) - \alpha_-)\mathbf{f}(x) & \text{otherwise} \end{cases}. \quad (6)$$

The feasibility of (5) is discussed in Appendix C. The proposed method is summarized in Alg. 3. The classifier-based Ω generalizes the validity-based Ω. First, any validity-based Ω can be represented by a classifier-based Ω if we let $\mathbf{f} = \mathbf{v}$ and $\tau = \frac{1}{2}$. Next, we note there is a trivial way to deal with classifier-based Ω via the validity-based approach – by setting $\mathbf{v}(x) = 1\{\mathbf{f}(x) < \tau\}$. However, potentially useful information such as values and gradients of $\mathbf{f}$ are lost, and we will evaluate this effect in experiments. In addition, the classifier-based approach does not maintain the potentially large set of invalid generated samples, as this step is automatically done in the `guide` function.

D. Generalization to Multiple Forgetting Sets

Let $\{\Omega_k\}_{k=1}^K$ be disjoint sets in $\mathbb{R}^d$, and we would like the model to forget $\Omega = \bigcup_{k=1}^K \Omega_k$. In the data-based setting, we let $p_\Omega = \mathcal{U}(\Omega) = \mathcal{U}(\bigcup_{k=1}^K \Omega_k)$. In the validity-based setting, each Ω_k is associated with a validity function $\mathbf{v}_k$. We let the overall validity function to be $\mathbf{v}(x) = \min_k \mathbf{v}_k(x)$. In the classifier-based setting, each Ω_k is associated with a classifier $\mathbf{f}_k$. Similar to the validity-based setting, we let the overall $\mathbf{f}$ to be $\mathbf{f}(x) = \min_k \mathbf{f}_k(x)$.

IV. Experiments

In this section, we aim to answer the following questions.

- How well can the algorithms in Section III forget samples in practice?
- Can these algorithms be used to de-bias pre-trained models?
- Can these algorithms be used to understand training data?

In this section, we examine these questions by focusing on several real-world image datasets, including MNIST (28×28) [LeCun et al., 2010], CIFAR (32×32) [Krizhevsky et al., 2009], CelebA (64×64) [Liu et al., 2015] and STL-10 (96×96) [Coates et al., 2011] datasets. In Section IV-A, we investigate how well these algorithms can forget samples with a specific label. In Section IV-B, we investigate how well these algorithms can de-bias pre-trained models and improve generation quality. In Section IV-C, we use these algorithms to understand training data through the lens of data forgetting.

The pre-trained model for each dataset is a DCGAN [Radford et al., 2015] trained for 200 epochs (see details in Appendix D). We use one

NVIDIA 3080 GPU to train these models and run experiments.

Evaluation Metrics: invalidity and generation quality. The invalidity is defined as the mass of the generation distribution on the forgetting set Ω: $\texttt{Inv}(p_G) = \int_{x\in\Omega} p_G(x)dx$. In practice, we measure invalidity by generating 50K samples and computing the fraction of these samples that fall into Ω.

The generation quality is measured in Inception Score (IS) [Salimans et al., 2016] and Frechet Inception Distance (FID) [Heusel et al., 2017]. Higher IS or lower FID indicates better quality. We compute IS for grey-scale images and FID for RGB images. When measuring quality, we compute IS or FID between 50K generated samples and $X \cap \bar{\Omega}$. Therefore, this score is not comparable with the score w.r.t. the pre-trained model if the forgetting set includes samples in the training set, such as samples with a specific label in Section IV-A. Detailed setup is in Appendix D.

A. Forgetting Labels

Question. How well can the algorithms in Section III forget samples in practice?

Methodology. We investigate how well the proposed algorithms can forget samples with a specific label y. In the data-based setting (Alg. 1), we express this as $\Omega = \{x \in X : \texttt{label}(x) = y\}$. In the validity-based setting (Alg. 2), we express this by setting $\mathbf{v}(x) = 1\{\arg\max_i \texttt{logit}(x)_i \neq y\}$, where logit is the output of the softmax layer of a pre-trained label classifier [Chen, 2020]. In the classifier-based setting (Alg. 3), we set $\mathbf{f}(x) = 1 - \texttt{logit}(x)_y$.

In Table I, we compare invalidity and generation quality among different algorithms and datasets when we forget label 0. We plot invalidity during data forgetting in Fig. 2. We also compare invalidity after one epoch of data forgetting in Appendix E-A2. Mean and standard errors for 5 random runs are reported. Results for different hyper-parameters and forgetting other labels are in Appendix E.

Results. We find all the algorithms in Section III work quite well with a much fewer number of epochs used for training the pre-trained model (which is 200). These algorithms are generally comparable. Therefore, we conclude that the simplest data-based algorithm is good enough to forget samples when those training samples to be forgotten ($X \cap \Omega$) can characterize the forgetting set (Ω) well.

We also find invalidity rapidly drops after only one epoch of data forgetting, indicating these algorithms are very efficient in penalizing invalidity. While different algorithms perform better on different datasets, they are highly comparable with each other. The reason why the classifier-based algorithm performs the best on MNIST is possibly that the label classifier on MNIST is almost perfect so its gradient information is accurate.

Visualization. We sample latents $z \sim \mathcal{N}(0, I)$ and choose those corresponding to invalid samples, i.e. $G_0(z) \in \Omega$ where G_0 is the pre-trained generator. We select visually good $G_0(z)$ for demonstration. We visualize $G(z)$ during data forgetting in Fig. 3, and more visualizations are in Appendix E-C. This demonstrates how the latent space is manipulated: the label to be forgotten is gradually pushed to other labels, and there is high-level visual similarity between the final $G(z)$ and the original $G_0(z)$.

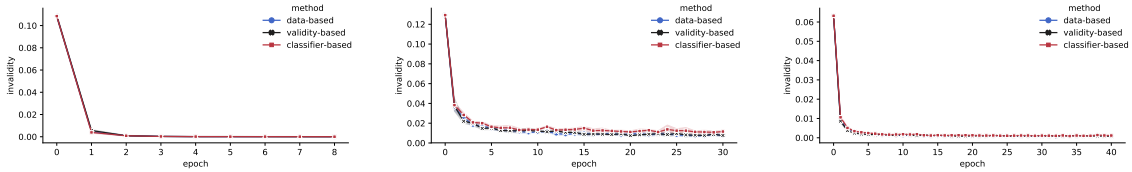
Effects of other hyper-parameters. In Table II, we compare different T (#queries after each epoch) in the validity-based forgetting algorithm (Alg. 2). We fix the total number of queries by setting $T\times$#epochs to be a constant. Results indicate that a large T may lead to worse invalidity, and there is trade-off between invalidity and quality when setting T to be small or moderate.

In Appendix E-A4, we compare different λ (hyperparameter in (2)) in the classifier-based forgetting algorithm (Alg. 3). We find there exists a clear trade-off between invalidity and quality when alternating λ: a larger λ tends to produce better quality, and a smaller λ tends to have better invalidity.

Forgetting multiple sets. We then investigate how well the proposed algorithms can generalize to multiple forgetting sets with methods in Section III-D. We focus on the CelebA dataset [Liu et al., 2015], which has 40 labeled attributes. We use proposed algorithms to forget a combination of these attributes: $\Omega_1 = \{\texttt{Black_hair and Blurry}\}$, $\Omega_2 = \{\texttt{Brown_hair and Wear_eyeglasses}\}$, and $\Omega = \Omega_1 \cup \Omega_2$. These attributes are randomly selected

TABLE I: Invalidity and generation quality of different forgetting algorithms on forgetting label zero within different datasets. Mean and standard errors are reported for five random seeds. Note that quality measure after data forgetting is not directly comparable with the pre-trained model. The invalidity drops in magnitude after data forgetting. Different forgetting algorithms are highly comparable to each other.

Dataset	Evaluation	Pre-trained	Data-based	Validity-based	Classifier-based
MNIST	Inv(↓)($\times 10^{-5}$)	1.1×10^4	8.0 ± 2.2	6.4 ± 0.8	$\mathbf{5.2 \pm 3.7}$
(8 epochs)	IS(↑)	7.82	$\mathbf{7.20 \pm 0.08}$	7.19 ± 0.04	7.16 ± 0.04
CIFAR-10	Inv(↓)($\times 10^{-3}$)	1.3×10^2	$\mathbf{7.5 \pm 1.1}$	7.6 ± 1.0	11.6 ± 1.0
(30 epochs)	FID(↓)	36.2	34.8 ± 1.5	34.8 ± 1.4	$\mathbf{33.2 \pm 0.6}$
STL-10	Inv(↓)($\times 10^{-4}$)	6.2×10^2	8.8 ± 4.5	$\mathbf{7.7 \pm 1.3}$	11.6 ± 3.6
(40 epochs)	FID(↓)	79.1	77.8 ± 2.2	$\mathbf{77.0 \pm 2.3}$	77.2 ± 1.5



(a) MNIST (b) CIFAR-10 (c) STL-10

Fig. 2: Invalidity during data forgetting when forgetting label zero. Mean and standard errors are plotted for five random seeds. Standard errors may be too small to spot. Invalidity drops quickly at the beginning of data forgetting, and different algorithms are highly comparable to each other.

TABLE II: Study on the effect of T in Alg. 2 when the total number of queries is fixed. R refers to the number of epochs of data forgetting. A large T may lead to worse invalidity.

T	MNIST			CIFAR-10			STL-10		
	R	Inv(↓)	IS(↑)	R	Inv(↓)	FID(↓)	R	Inv(↓)	FID(↓)
400	20	$\mathbf{0.0 \times 10^{-4}}$	7.10	75	$\mathbf{0.45 \times 10^{-2}}$	35.1	100	1.0×10^{-3}	**75.1**
1000	8	0.6×10^{-4}	**7.19**	30	0.76×10^{-2}	34.8	40	$\mathbf{0.8 \times 10^{-3}}$	77.0
2000	4	2.8×10^{-4}	7.11	15	1.00×10^{-2}	**31.9**	20	1.0×10^{-3}	**75.1**

from those easy to capture. See detailed setup in Appendix E-D. Results after 1 or 5 epochs are reported in Table III. Consistent with results on forgetting just one label, all algorithms can reduce invalidity and retain generation quality and are comparable, while the classifier-based algorithm achieves the best invalidity after one epoch.

B. Model De-biasing

There can be different artifacts in GAN generated samples, and these could harm the overall generation quality. These artifacts may not exist in training samples, but are caused by inductive biases of the model, and become obvious after training. We can post-edit a pre-trained model to remove these artifacts, which we call *model de-biasing*. In this section, we investigate how well Alg. 2 and Alg. 3 apply to this task. We assume training samples are not biased so Alg. 1 does not apply to de-biasing.

To use these algorithms for de-biasing, we assume the target artifact or bias can be automatically detected by a classifier $\mathbf{f}$ or a validity function $\mathbf{v}$. Specifically, we survey two kinds of biases: boundary artifacts and label biases.

Boundary artifacts. A GAN trained on MNIST might generate samples that have numerous white pixels on the boundary (see Appendix F-A). We call this phenomenon the *boundary artifact*. We use the validity-based algorithm (Alg. 2) to de-bias boundary artifacts. The validity function is defined as

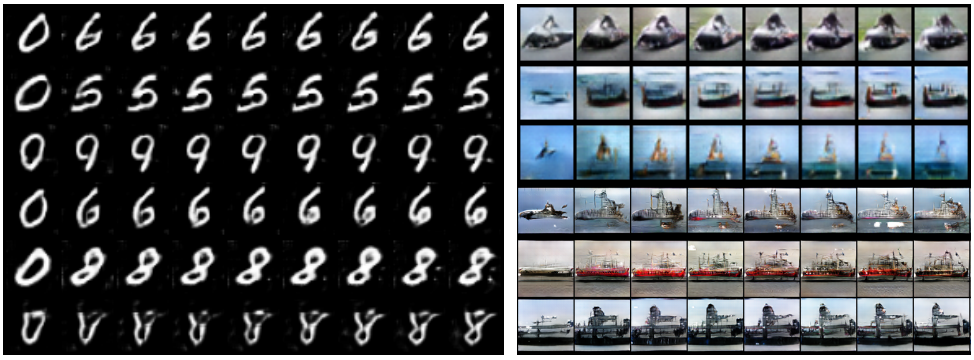


Fig. 3: Visualization of the data forgetting process of invalid samples when forgetting label zero. The first column is generated by the pre-trained generator, and the i-th column is generated after $k \cdot (i-1)$ epochs of data forgetting. Left: MNIST with $k = 1$. Right: top is CIFAR-10 and bottom is STL-10, both with $k = 4$ and label zero being `airplanes`. We can see samples associated with invalid labels are gradually pushed to other labels, but a high-level visual similarity is kept.

TABLE III: Invalidity and generation quality of different forgetting algorithms on forgetting a combination of attributes within CelebA. There is a significant drop of invalidity, indicating that different forgetting algorithms can all generalize to multiple forgetting sets.

Evaluation	Pre-trained	Epochs	Data-based	Validity-based	Classifier-based
`Inv`(↓)	1.66×10^{-3}	1	9.0×10^{-4}	7.6×10^{-4}	$\mathbf{7.0 \times 10^{-4}}$
`Inv`(↓)	1.66×10^{-3}	5	$\mathbf{3.8 \times 10^{-4}}$	6.8×10^{-4}	6.8×10^{-4}
`FID`(↓)	36.4	5	29.3	29.9	**27.9**

$\mathbf{v}(x) = 1\{\sum_{(i,j)\in\text{boundary pixels}} x_{ij} < \tau_b\}$, where boundary pixels are those within a certain margin to the boundary, and threshold τ_b satisfies no training image is invalid.

Results are reported in Table IV. It is clear that the invalidity reduces in order after data forgetting, indicating boundary artifacts are largely removed. Consistent with Table II, a small or moderate T leads to better results. We visualize samples before and after de-biasing in Appendix F-A.

Label biases. Neural networks may generate visually smooth but semantically ambiguous samples [Kirichenko et al., 2020], e.g. samples that look like multiple objects (see Appendix F-B). We call this phenomenon the *label bias*. We use the classifier-based algorithm (Alg. 3) to de-bias label biases. The classifier is defined as $\mathbf{f}(x) = 1 - \text{Entropy}(\texttt{logit}(x))/\log(\texttt{\#classes})$, where the `logit` function is the same as in Section IV-A. We also compare to a data deletion baseline, where we delete invalid samples and fully re-train the model. Results are reported in Table V and VI. After de-biasing, we can improve the generation quality by a significant gap (~ 0.3 in `IS` and ~ 10 in `FID`). There is also a clear drop in terms of invalidity. In contrast, we find that data deletion does not help removing label biases.

C. Understanding Training Data through the Lens of Data Forgetting

Large datasets can be hard to analyze. In this section, we investigate how data forgetting can help us understand these data. Specifically, we ask: which samples are easy or hard to forget?

In order to quantify the difficulty to forget a sample, we define the forgetting score $\mathcal{FS}$ to be the difference of discriminator outputs before and after data forgetting. Formally, let $x \in \Omega$ be a sample to forget, $\mathcal{M} = (G_0, D_0)$ be the pre-trained model, and $\mathcal{M}' = (G', D')$ be a model

TABLE IV: Invalidity after de-biasing boundary artifacts of generated MNIST samples. We run the validity-based forgetting algorithm (Alg. 2) for 4 epochs. The invalidity drops significantly, and a small or moderate T leads to slightly lower (better) invalidity.

	Pre-trained	$T = 5$K	$T = 10$K	$T = 20$K	$T = 40$K	$T = 80$K
Margin $= 1$	3.1×10^{-3}	$\mathbf{6.0 \times 10^{-5}}$	8.0×10^{-5}	2.0×10^{-4}	2.0×10^{-4}	7.0×10^{-4}
Margin $= 2$	1.1×10^{-3}	1.6×10^{-4}	$\mathbf{4.0 \times 10^{-5}}$	6.0×10^{-5}	3.2×10^{-4}	2.8×10^{-4}

TABLE V: Invalidity and Inception scores after de-biasing label biases of generated samples from MNIST. We run the classifier-based forgetting algorithm (Alg. 3) for 8 epochs with $\lambda = 0.8$, and compare to the data deletion baseline with 200 epochs of full re-training. The arrow means improvement from the pre-trained model to after data forgetting. There is a clear improvement of generation quality, indicating the proposed algorithm can help GANs generate better samples. In contrast, data deletion does not help improve invalidity or quality.

τ	Forgetting (8 epochs)		Data deletion baseline (200 epochs)	
	Inv(↓)	IS(↑)	Inv(↓)	IS(↑)
0.3	$8.19 \times 10^{-4} \to 2.60 \times 10^{-4}$	$7.82 \to \mathbf{8.10}$	$8.19 \times 10^{-4} \to 1.14 \times 10^{-3}$	$7.82 \to 7.75$
0.5	$2.07 \times 10^{-2} \to 1.70 \times 10^{-2}$	$7.82 \to 7.92$	$2.07 \times 10^{-2} \to 2.17 \times 10^{-2}$	$7.82 \to 7.79$
0.7	$1.35 \times 10^{-1} \to 1.22 \times 10^{-1}$	$7.82 \to 7.95$	$1.35 \times 10^{-1} \to 1.32 \times 10^{-1}$	$7.82 \to 7.82$

TABLE VI: Invalidity and FID scores after de-biasing label biases of generated samples from CIFAR-10. We run the classifier-based forgetting algorithm (Alg. 3) for 30 epochs with $\lambda = 0.9$. The arrow means improvement from the pre-trained model to after data forgetting. There is a clear improvement of generation quality, indicating the proposed algorithm can help GANs generate better samples. Note that there is **no** invalid sample in the training set, so the data deletion baseline is identical to the pre-trained model.

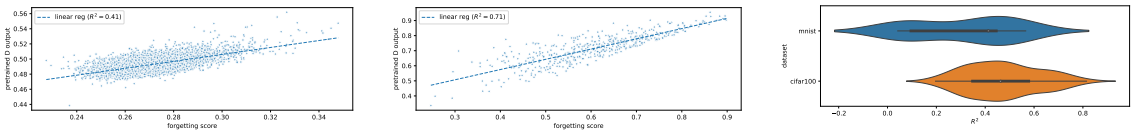
τ	Inv(↓)	FID(↓)
0.5	$2.28 \times 10^{-2} \to 1.67 \times 10^{-2}$	$36.2 \to \mathbf{26.6}$
0.7	$1.72 \times 10^{-1} \to 1.49 \times 10^{-1}$	$36.2 \to 26.8$
0.3	$5.79 \times 10^{-4} \to 2.20 \times 10^{-4}$	$36.2 \to 27.1$

after data forgetting. Then, the forgetting score is $\mathcal{FS}(x) = D_0(x) - D'(x)$. A larger $\mathcal{FS}$ means it is easier to forget x.

To investigate **sample-level** forgetting difficulty, we forget a particular label at one time using Alg. 1. We then demonstrate scatter plots of forgetting scores $\mathcal{FS}(x)$ versus pre-trained discriminator outputs $D_0(x)$ for all samples x with this label. We also fit linear regression and report R^2 values (larger means stronger linear relationship). Scatter plots for some labels in MNIST and CIFAR-100 and distribution of R^2 for all labels are shown in Fig. 4. We also visualize the most and least difficult-to-forget samples in Appendix G. We find there is positive correlation between $\mathcal{FS}(x)$ and $D_0(x)$, indicating on-manifold (large $D_0(x)$) samples are easier to be forgotten, while off-manifold (small $D_0(x)$) ones are harder to be forgotten. This analysis further provides a way to investigate **label-level** forgetting difficulty. By averaging forgetting scores for samples associated with each label, we can survey which labels are easy or hard to forget in general. The results are in Appendix G. We find some labels are harder to forget than others.

D. Discussion: Relationship to Adversarial Samples

An adversarial sample for a classifier $\mathbf{f}$ and a sample x is another sample $\tilde{x} \approx x$ but $\mathbf{f}(\tilde{x}) \neq \mathbf{f}(x)$. Generating and defending these samples have become one of the most important directions of deep learning [Goodfellow et al., 2014b, Madry et al., 2018]. In this section, we show a variant of Alg. 3 can potentially be used to define a specific type of



(a) MNIST (0) (b) CIFAR-100 (`aquarium_fish`) (c) Distribution of R^2

Fig. 4: (a) and (b) Forgetting scores of invalid training samples ($\mathcal{FS}(x)$) versus the pre-trained discriminator outputs of them ($D_0(x)$). There is positive correlation between these two scores, indicating on-manifold samples are easier to forget. (a) Forgetting 0 in MNIST. (b) Forgetting `aquarium_fish` in CIFAR-100. (c) Distributions of R^2 scores of linear regression between $\mathcal{FS}(x)$ and $D_0(x)$ for all labels. The correlation in (a) and (b) is universal and stronger in CIFAR-100.

adversarial samples. In detail, we fix the discriminator D and only update the generator G while running Alg. 3. Then, G is trained to fool D and the classifier $\mathbf{f}$ at the same time. Notice that fooling D means generating on-manifold (visually similar to training data) samples, and fooling $\mathbf{f}$ means finding adversarial samples of $\mathbf{f}$. By combining these objectives we can force G to produce on-manifold adversarial samples, which may be significant in many real-world applications. We visualize some samples in Appendix H.

V. Related Work

Although deep generative models have been highly successful at many domains, it has long been known that they often emit undesirable samples and samples with different types of artifacts that make them untrustworthy. Examples include blurred image samples [Kaneko and Harada, 2021], fairness issues [Tan et al., 2020, Karakas et al., 2022], and checkerboard artifacts [Odena et al., 2016, Zhang et al., 2019, Wang et al., 2020, Schwarz et al., 2021] in image generation, offensive text in language models [Abid et al., 2021, Perez et al., 2022], and unnatural sound in speech models [Donahue et al., 2018, Thiem et al., 2020].

Some prior works have used post-editing to remove artifacts and improve GANs. Examples include improving fairness [Tan et al., 2020, Karakas et al., 2022], rule rewriting [Bau et al., 2020], discovering interpretability [Härkönen et al., 2020], and fine-tuning [Mo et al., 2020, Li et al., 2020, Zhao et al., 2020]. The purpose, use cases, and editing methods of these papers are different from our paper, where we focus on data forgetting.

While our problem definition and formalization is novel, the technical solutions that we propose are related to three prior works that use these techniques in different contexts. These are NDA [Sinha et al., 2020], Rumi-GAN [Asokan and Seelamantula, 2020], and Hanneke et al. [2018]. The first two papers look at how to avoid generating negative samples while training a generative model from scratch. This is done by defining new fake distributions to penalize the generation of these samples. However, their purposes are different from us: NDA is used to characterize the boundary of the support of the generative distribution more precisely, and Rumi-GAN is used to handle unbalanced data. We extend their idea and theory to data forgetting in Section III. [1] Hanneke et al. [2018] propose an active learning approach to avoid generating invalid samples, also while training a generative model from scratch. Their work however is entirely theoretical and apply to discrete distributions. In our paper, the validity-based forgetting algorithm (Alg. 2) is based on a simplified version of their algorithm. We also use their definition of *invalidity* as an evaluation method.

Our work is also related to data deletion or machine unlearning [Cao and Yang, 2015, Guo et al., 2019, Schelter, 2020, Neel et al., 2021, Sekhari et al., 2021, Izzo et al., 2021, Ullah et al., 2021]. However, there are two important differences between data

[1]The loss functions in NDA and Rumi-GAN are similar.

deletion and data forgetting. First, data deletion aims to approximate the re-trained model when some training samples are removed – mostly due to privacy reasons – while in data forgetting we penalize the model from knowing samples that should be forgotten. Another difference is that in data forgetting, the forgetting set Ω may have a zero intersection with training data. These two differences are discussed in Section II-C in detail. In addition, most data deletion techniques are for supervised learning or clustering, and is much less studied for generative models.

There is also a related line of work on catastrophic forgetting in supervised learning [Kirkpatrick et al., 2017] and generative models [Thanh-Tung and Tran, 2020]. This concept is different from data forgetting in that we would like the generative model to forget certain data after training, while catastrophic forgetting means knowledge learned in previous tasks is destroyed during continual learning.

VI. Conclusion

In this paper, we propose a systematic framework for forgetting data from pre-trained generative models. We provide three different algorithms for GANs that differ on how the samples to be forgotten are described. We provide theoretical results that data forgetting can be achieved. We then empirically investigate data forgetting on real-world image datasets, and show that our algorithms are capable of forgetting data while retaining high generation quality at a fraction of the cost of full re-training. One limitation or our paper is that the proposed framework only applies to unconditional generative models. It is an important future direction to define data forgetting and propose algorithms for conditional generative models, which are more widely used in downstream deep learning applications.

References

A. Abid, M. Farooqi, and J. Zou. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306, 2021.

A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*, 2017.

S. Asokan and C. Seelamantula. Teaching a gan what not to learn. *Advances in Neural Information Processing Systems*, 33:3964–3975, 2020.

D. Bau, S. Liu, T. Wang, J.-Y. Zhu, and A. Torralba. Rewriting a deep generative model. In *European Conference on Computer Vision*, pages 351–369. Springer, 2020.

Y. Cao and J. Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

A. Chen. Pytorch-playground. https://github.com/aaron-xichen/pytorch-playground, 2020.

A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

C. de Masson d'Autume, S. Mohamed, M. Rosca, and J. Rae. Training language gans from scratch. *Advances in Neural Information Processing Systems*, 32, 2019.

C. Donahue, J. McAuley, and M. Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014a.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explain-

ing and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

S. Hanneke, A. T. Kalai, G. Kamath, and C. Tzamos. Actively avoiding nonsense in generative models. In *Conference On Learning Theory*, pages 209–227. PMLR, 2018.

E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850, 2020.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

W. R. Inc. Mathematica, Version 13.0.0. URL https://www.wolfram.com/mathematica. Champaign, IL, 2021.

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021.

T. Kaneko and T. Harada. Blur, noise, and compression robust generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13579–13589, 2021.

C. Karakas, A. Dirik, E. Yalcinkaya, and P. Yanardag. Fairstyle: Debiasing stylegan2 with style channel manipulations. *arXiv preprint arXiv:2202.06240*, 2022.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

P. Kirichenko, P. Izmailov, and A. G. Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33:20578–20589, 2020.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Q. Li, L. Mai, M. A. Alcorn, and A. Nguyen. A cost-effective method for improving and re-purposing large, pre-trained gans by fine-tuning their class-embeddings. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

S. Mo, M. Cho, and J. Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020.

S. Neel, A. Roth, and S. Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine

unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR, 2021.

X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

E. Perez, S. Huang, F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

G. K. Pitsilis, H. Ramampiaro, and H. Langseth. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*, 2018.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

S. Schelter. " amnesia"-machine learning models that can forget user data very fast. In *CIDR*, 2020.

K. Schwarz, Y. Liao, and A. Geiger. On the frequency bias of generative models. *Advances in Neural Information Processing Systems*, 34, 2021.

A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34, 2021.

A. Sinha, K. Ayush, J. Song, B. Uzkent, H. Jin, and S. Ermon. Negative data augmentation. In *International Conference on Learning Representations*, 2020.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

A. Taha, A. Shrivastava, and L. S. Davis. Knowledge evolution in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12843–12852, 2021.

S. Tan, Y. Shen, and B. Zhou. Improving the fairness of deep generative models without retraining. *arXiv preprint arXiv:2012.04842*, 2020.

H. Thanh-Tung and T. Tran. Catastrophic forgetting and mode collapse in gans. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2020.

N. Thiem, M. Orescanin, and J. B. Michael. Reducing artifacts in gan audio synthesis. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1268–1275. IEEE, 2020.

S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

E. Ullah, T. Mai, A. Rao, R. A. Rossi, and R. Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pages 4126–4142. PMLR, 2021.

S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8695–8704, 2020.

D. Warde-Farley and I. Goodfellow. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, 311:5, 2016.

J. Zhang, X. Chen, Z. Cai, L. Pan, H. Zhao, S. Yi, C. K. Yeo, B. Dai, and C. C. Loy. Unsupervised 3d shape completion through gan inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1768–1777, 2021.

X. Zhang, S. Karaman, and S.-F. Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6.

IEEE, 2019.

M. Zhao, Y. Cong, and L. Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning*, pages 11340–11351. PMLR, 2020.

H. Zhou, A. Vani, H. Larochelle, and A. Courville. Fortuitous forgetting in connectionist networks. In *International Conference on Learning Representations*, 2021.

J. Zhu, Y. Shen, D. Zhao, and B. Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020.

Appendix A
Proof of Theorem 1 and Extension to f-GAN

a) Background of f-GAN [Nowozin et al., 2016].: Let ϕ be a convex, lower-semicontinuous function such that $\phi(1) = 0$. In f-GAN, the following ϕ-divergence is minimized:

$$D_\phi(P\|Q) = \int_{x\in\mathbb{R}^d} Q(x)\phi\left(\frac{P(x)}{Q(x)}\right) dx.$$

According to the variational characterization of ϕ-divergence [Nguyen et al., 2010],

$$D_\phi(P\|Q) = \sup_T \left[\mathbb{E}_{x\sim P} T(x) - \mathbb{E}_{x\sim Q}\phi^*(T(x))\right],$$

where the optimal T is obtained by $T = \phi'\left(\frac{P}{Q}\right)$.

b) The objective function (3) corresponds to an f-GAN.: Let $\alpha = \alpha_- + \alpha_+$. We can rewrite (3) as

$$L(G, D) = \alpha \cdot \mathbb{E}_{x\sim P} \log D(x) + (2-\alpha)\cdot \mathbb{E}_{x\sim Q}\log(1 - D(x)),$$

where

$$P = \frac{\alpha_+}{\alpha} p_{\text{data}}|_{\bar{\Omega}} + \frac{\alpha_-}{\alpha} p_{\text{fake}}; \quad Q = \frac{1-\alpha_+}{2-\alpha} p_{\text{data}}|_{\bar{\Omega}} + \frac{1-\alpha_-}{2-\alpha} p_{\text{fake}}.$$

Let

$$C = \alpha\log\alpha + (2-\alpha)\log(2-\alpha) - 2\log 2,$$

$$\phi(u) = (\alpha u)\log(\alpha u) - (\alpha u - \alpha + 2)\log(\alpha u - \alpha + 2) + (2-\alpha)\log(2-\alpha) - C.$$

Then, $\phi(1) = 0$, and $\phi''(u) = \frac{\alpha(2-\alpha)}{u(\alpha u - \alpha + 2)} > 0$ so ϕ is convex. Its convex conjugate function ϕ^* is

$$\phi^*(t) := \sup_u (ut - \phi(u)) = -(2-\alpha)\log\left(1 - e^{\frac{t}{\alpha}}\right) + C.$$

Let $T(x) = \alpha \log D(x)$. Then,

$$\max_D L(G, D) = \sup_T \left[\mathbb{E}_{x\sim P} T(x) - \mathbb{E}_{x\sim Q}\phi^*(T(x))\right] + C = D_\phi(P\|Q) + C.$$

c) Optimal D.: We have

$$\phi'(u) = \alpha \log \frac{\alpha u}{\alpha u - \alpha + 2}.$$

Therefore, the optimal discriminator is

$$\alpha \log D = \phi'\left(\frac{P}{Q}\right),$$

or

$$D = \frac{\alpha P}{\alpha P + (2-\alpha)Q} = \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_- p_{\text{fake}}}{p_{\text{data}}|_{\bar{\Omega}} + p_{\text{fake}}}.$$

Finally, the optimal discriminator in (4) is obtained by inserting (2) into the above equation.

d) Optimal G.: For conciseness, we let

$$P_1 = p_{\text{data}}|_{\bar{\Omega}}, P_2 = p_G, P_3 = p_\Omega,$$
$$\beta_1 = \frac{\alpha_+}{\alpha}, \beta_2 = \frac{\alpha_-\lambda}{\alpha}, \beta_3 = \frac{\alpha_-(1-\lambda)}{\alpha},$$
$$\gamma_1 = \frac{1-\alpha_+}{2-\alpha}, \gamma_2 = \frac{(1-\alpha_-)\lambda}{2-\alpha}, \gamma_3 = \frac{(1-\alpha_-)(1-\lambda)}{2-\alpha}.$$

Then, we have

$$P = \sum_{i=1}^{3} \beta_i P_i, Q = \sum_{i=1}^{3} \gamma_i P_i.$$

We also have

$$\frac{\beta_1}{\gamma_1} > \frac{\beta_2}{\gamma_2} = \frac{\beta_3}{\gamma_3}.$$

Because $\mathbf{supp}(P_1) \cap \mathbf{supp}(P_3)$ is the empty set, we have

$$\begin{aligned} D_\phi(P\|Q) &= \int_{x\in\mathbb{R}^d} \left(\sum_{i=1}^{3} \gamma_i P_i\right) \phi\left(\frac{\sum_{i=1}^{3} \beta_i P_i}{\sum_{i=1}^{3} \gamma_i P_i}\right) dx \\ &= \int_{x\notin\Omega} (\gamma_1 P_1 + \gamma_2 P_2)\phi\left(\frac{\beta_1 P_1 + \beta_2 P_2}{\gamma_1 P_1 + \gamma_2 P_2}\right) dx \\ &+ \int_{x\in\Omega} (\gamma_2 P_2 + \gamma_3 P_3)\phi\left(\frac{\beta_2 P_2 + \beta_3 P_3}{\gamma_2 P_2 + \gamma_3 P_3}\right) dx \end{aligned}$$

Let

$$\int_{x\in\Omega} P_2 dx = \eta.$$

We have

$$\int_{x\in\Omega} (\gamma_2 P_2 + \gamma_3 P_3)\phi\left(\frac{\beta_2 P_2 + \beta_3 P_3}{\gamma_2 P_2 + \gamma_3 P_3}\right) dx = (\gamma_2\eta + \gamma_3)\,\phi\left(\frac{\beta_3}{\gamma_3}\right).$$

Let

$$\zeta = \frac{\beta_2(\gamma_1+\gamma_2)}{\gamma_2(\beta_1+\beta_2)}.$$

According to Jensen's inequality,

$$\begin{aligned} &\int_{x\notin\Omega} (\gamma_1 P_1 + \gamma_2 P_2)\phi\left(\frac{\beta_1 P_1 + \beta_2 P_2}{\gamma_1 P_1 + \gamma_2 P_2}\right) dx \\ &= (\gamma_1 + \gamma_2(1-\zeta\eta)) \int_{x\notin\Omega} \left(\frac{\gamma_1 P_1 + \gamma_2 P_2}{\gamma_1 + \gamma_2(1-\zeta\eta)}\right) \phi\left(\frac{\beta_1 P_1 + \beta_2 P_2}{\gamma_1 P_1 + \gamma_2 P_2}\right) dx \\ &\geq (\gamma_1 + \gamma_2(1-\zeta\eta))\phi\left(\int_{x\notin\Omega} \frac{\beta_1 P_1 + \beta_2 P_2}{\gamma_1 + \gamma_2(1-\zeta\eta)} dx\right) \\ &= (\gamma_1 + \gamma_2(1-\zeta\eta))\phi\left(\frac{\beta_1 + \beta_2(1-\eta)}{\gamma_1 + \gamma_2(1-\zeta\eta)}\right) \\ &= (\gamma_1 + \gamma_2(1-\zeta\eta))\phi\left(\frac{\beta_1 + \beta_2}{\gamma_1 + \gamma_2}\right). \end{aligned}$$

Therefore, we have

$$D_\phi(P\|Q) \geq (\gamma_1+\gamma_2)\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right) + \gamma_3\phi\left(\frac{\beta_3}{\gamma_3}\right) + \left[\gamma_2\phi\left(\frac{\beta_3}{\gamma_3}\right) - \frac{\beta_2(\gamma_1+\gamma_2)}{\beta_1+\beta_2}\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right)\right]\eta.$$

Now, we show the η term is non-negative. We write

$$\begin{aligned}\gamma_2\phi\left(\frac{\beta_3}{\gamma_3}\right) - \frac{\beta_2(\gamma_1+\gamma_2)}{\beta_1+\beta_2}\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right) &= \beta_2\left(\frac{\gamma_2}{\beta_2}\phi\left(\frac{\beta_3}{\gamma_3}\right) - \frac{(\gamma_1+\gamma_2)}{\beta_1+\beta_2}\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right)\right)\\ &= \beta_2\left(\frac{\gamma_3}{\beta_3}\phi\left(\frac{\beta_3}{\gamma_3}\right) - \frac{1-\gamma_3}{1-\beta_3}\phi\left(\frac{1-\beta_3}{1-\gamma_3}\right)\right).\end{aligned}$$

It suffices to prove the function $\psi(u) = \phi(u)/u$ satisfies

$$\psi\left(\frac{\beta_3}{\gamma_3}\right) \geq \psi\left(\frac{1-\beta_3}{1-\gamma_3}\right).$$

We use the Mathematica software [Inc.] to compute the difference:

$$\begin{aligned}\psi\left(\frac{\beta_3}{\gamma_3}\right) - \psi\left(\frac{1-\beta_3}{1-\gamma_3}\right) = &-\frac{\alpha}{\alpha_-}\log\frac{2-\alpha}{1-\alpha_-} + \alpha\alpha_-\log\frac{\alpha_-(2-\alpha)}{1-\alpha_-} + \frac{\alpha(1-\alpha_-)}{2-\alpha}(\log 4 - \alpha\log\alpha)\\ &-\frac{\alpha}{2-\alpha}\left(\frac{\lambda+1}{\lambda\alpha_-+\alpha_+} - 1\right)(\log 4 - \alpha\log\alpha)\\ &-\alpha\log\frac{(2-\alpha)(\lambda\alpha_-+\alpha_+)}{\lambda(1-\alpha_-)+1-\alpha_+} + \frac{\alpha(\lambda+1)}{\lambda\alpha_-+\alpha_+}\log\frac{(\lambda+1)(2-\alpha)}{\lambda(1-\alpha_-)+1-\alpha_+}.\end{aligned}$$

The minimum value of the above difference for $\alpha_- \in [0, \frac{1}{2}]$, $\alpha_+ \in [0, \frac{1}{2}]$, and $\lambda \in [0, 1]$ is obtained at $\alpha_- = \alpha_+ = \frac{1}{2}$, where the difference equals zero. This makes us able to conclude

$$D_\phi(P\|Q) \geq (\gamma_1+\gamma_2)\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right) + \gamma_3\phi\left(\frac{\beta_3}{\gamma_3}\right).$$

Finally, we let $P_2 = P_1$. In this case,

$$\begin{aligned}D_\phi(P\|Q) &= \int_{x\in\mathbb{R}^d}\left(\sum_{i=1}^3\gamma_iP_i\right)\phi\left(\frac{\sum_{i=1}^3\beta_iP_i}{\sum_{i=1}^3\gamma_iP_i}\right)dx\\ &= \int_{x\notin\Omega}(\gamma_1P_1+\gamma_2P_2)\phi\left(\frac{\beta_1P_1+\beta_2P_2}{\gamma_1P_1+\gamma_2P_2}\right)dx\\ &+ \int_{x\in\Omega}\gamma_3P_3\phi\left(\frac{\beta_3P_3}{\gamma_3P_3}\right)dx\\ &= (\gamma_1+\gamma_2)\phi\left(\frac{\beta_1+\beta_2}{\gamma_1+\gamma_2}\right) + \gamma_3\phi\left(\frac{\beta_3}{\gamma_3}\right).\end{aligned}$$

Therefore, the optimal generator is $p_G = p_{\text{data}}|_{\bar{\Omega}}$.

e) Extension to f-GAN.: We can extend the objective (3) to any type of f-GAN. Let ϕ be a convex, lower-semicontinuous function such that $\phi(1) = 0$. Let

$$P = \frac{\alpha_+}{\alpha}p_{\text{data}}|_{\bar{\Omega}} + \frac{\alpha_-}{\alpha}p_{\text{fake}};\quad Q = \frac{1-\alpha_+}{2-\alpha}p_{\text{data}}|_{\bar{\Omega}} + \frac{1-\alpha_-}{2-\alpha}p_{\text{fake}}.$$

We jointly optimize

$$\min_G\max_D\ L(G,D) = \mathbb{E}_{x\sim P}D(x) - \mathbb{E}_{x\sim Q}\phi^*(D(x)).$$

Then, the optimal discriminator is $D = \phi'\left(\frac{P}{Q}\right)$. If $\psi\left(\frac{\beta_3}{\gamma_3}\right) \geq \psi\left(\frac{1-\beta_3}{1-\gamma_3}\right)$, then the optimal generator is $p_G = p_{\text{data}}|_{\bar{\Omega}}$.

Remark 1. *When $\alpha_- = 0$ and $\alpha_+ = 1$ (i.e. there is no label smoothing), Theorem 1 in Sinha et al. [2020] implies the above optimal generator. Our theorem also extends their theorem to the label smoothing setting.*

APPENDIX B
THEORETICAL ANALYSIS OF A SIMPLIFIED DYNAMICAL SYSTEM ON INVALIDITY

In this section, we provide theoretical analysis to a simplified, ideal dynamical system that corresponds to Alg. 2 and Section III-B. In this dynamical system, we assume there are only two types of invalid samples: those easy to forget, and those hard to forget. We assume after each iteration, the generator will generate a less but positive fraction of invalid samples. Formally, let $\{\Omega_{\text{easy}}, \Omega_{\text{hard}}\}$ be a split of Ω, where Ω_{easy} is the set of invalid samples that are easy to forget, and Ω_{hard} is the set of invalid samples that are hard to forget. We let

$$\begin{aligned} m_{\text{easy}} &= \int_{\Omega_{\text{easy}}} p_G(x)dx, \\ m_{\text{hard}} &= \int_{\Omega_{\text{hard}}} p_G(x)dx, \\ m_{\text{ratio}} &= \frac{m_{\text{easy}}}{m_{\text{easy}} + m_{\text{hard}}}. \end{aligned}$$

Then, m_{easy} is the fraction of invalid generated samples that are easy to forget, and m_{hard} is the fraction of invalid generated samples that are hard to forget. $m_{\text{easy}} + m_{\text{hard}}$ is the fraction of invalid generated samples over all generated ones, which we call **invalidity**. We use superscript to represent each iteration. We consider the following dynamical system:

$$\begin{aligned} m_{\text{easy}}^{i+1} &= m_{\text{easy}}^{i} \cdot \eta_{\text{easy}}(m_{\text{ratio}}^{i}, T), \\ m_{\text{hard}}^{i+1} &= m_{\text{hard}}^{i} \cdot \eta_{\text{hard}}(m_{\text{ratio}}^{i}, T). \end{aligned}$$

In other words, the improvement of m_{easy} and m_{hard} (in terms of multiplication factor) is only affected by m_{ratio} and T. We make this assumption because in practice, the number of invalid samples to optimize the loss function is always fixed. As for boundary conditions, we assume $m_{\text{easy}}^{0} > m_{\text{hard}}^{0}$. We assume for $\eta \in \{\eta_{\text{easy}}, \eta_{\text{hard}}\}$, $0 < \eta(m, T) \leq 1$, where equality holds only in these situations:

$$\eta(m, 0) = 1, \ \eta_{\text{easy}}(0, T) = 1, \ \eta_{\text{hard}}(1, T) = 1.$$

We also assume a larger T leads to smaller η, but this effect degrades as T increases:

$$\frac{\partial}{\partial T}\eta(m, T) < 0, \ \frac{\partial^2}{\partial T^2}\eta(m, T) > 0.$$

To distinguish between samples that are easy or hard to forget, we assume

$$\frac{1}{m} \cdot \frac{\partial}{\partial T}\eta_{\text{easy}}(m, T) < \frac{1}{1-m} \cdot \frac{\partial}{\partial T}\eta_{\text{hard}}(m, T) < 0.$$

We can now draw some conclusions below.

a) As $i \to \infty$, invalidity converges to 0.: Because $\eta_{\text{easy}}(T) < 1$ and $\eta_{\text{hard}}(T) < 1$ when $T > 0$, we have $m_{\text{easy}}^{i+1} \leq m_{\text{easy}}^{i}$ and $m_{\text{hard}}^{i+1} \leq m_{\text{hard}}^{i}$. According to the monotone convergence theorem, there exists $m_{\text{easy}}^{\infty} \geq 0$ and $m_{\text{hard}}^{\infty} \geq 0$ such that

$$\lim_{i\to\infty} m_{\text{easy}}^{i} = m_{\text{easy}}^{\infty}, \ \lim_{i\to\infty} m_{\text{hard}}^{i} = m_{\text{hard}}^{\infty}.$$

We now prove $m_{\text{easy}}^{\infty} = m_{\text{hard}}^{\infty} = 0$. If otherwise, there exists $m_{\text{ratio}}^{\infty} = \frac{m_{\text{easy}}^{\infty}}{m_{\text{easy}}^{\infty} + m_{\text{hard}}^{\infty}}$ such that $m_{\text{ratio}}^{i} \to m_{\text{ratio}}^{\infty}$. We then have

$$\begin{aligned} m_{\text{easy}}^{\infty} &= m_{\text{easy}}^{\infty} \cdot \eta_{\text{easy}}(m_{\text{ratio}}^{\infty}, T), \\ m_{\text{hard}}^{\infty} &= m_{\text{hard}}^{\infty} \cdot \eta_{\text{hard}}(m_{\text{ratio}}^{\infty}, T). \end{aligned}$$

If $m^{\infty}_{\text{easy}} > 0$, then $m^{\infty}_{\text{ratio}} > 0$, and $\eta_{\text{easy}}(m^{\infty}_{\text{ratio}}, T) < 1$, contradiction. Similarly, if $m^{\infty}_{\text{hard}} > 0$, then $m^{\infty}_{\text{ratio}} < 1$, and $\eta_{\text{hard}}(m^{\infty}_{\text{ratio}}, T) < 1$, contradiction. Therefore, we conclude both m^{i}_{easy} and m^{i}_{hard} converge to 0. This indicates the invalidity converges to zero.

b) Simplifying the dynamical system.: To further simplify the problem, we make a strong assumption that η is linear in m. Then, we must have

$$\eta_{\text{easy}}(m, T) = 1 - \xi_{\text{easy}}(T) \cdot m,$$
$$\eta_{\text{hard}}(m, T) = 1 - \xi_{\text{hard}}(T) \cdot (1 - m),$$

where $\xi \in [0, 1], \xi(0) = 0, \xi' > 0, \xi'' < 0$ for $\xi \in \{\xi_{\text{easy}}, \xi_{\text{hard}}\}$. We also have $\xi'_{\text{easy}} > \xi'_{\text{hard}}$ and therefore $\xi_{\text{easy}} > \xi_{\text{hard}}$.

c) Optimal T and R from bounds.: We have

$$m^{i+1}_{\text{easy}} + m^{i+1}_{\text{hard}} = m^{i}_{\text{easy}} + m^{i}_{\text{hard}} - \frac{\xi_{\text{easy}}(T)(m^{i}_{\text{easy}})^2 + \xi_{\text{hard}}(T)(m^{i}_{\text{hard}})^2}{m^{i}_{\text{easy}} + m^{i}_{\text{hard}}}.$$

Because $\xi_{\text{easy}}(T) \geq \xi_{\text{hard}}(T)$, we have

$$\frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}(m^{i}_{\text{easy}} + m^{i}_{\text{hard}}) \leq \frac{\xi_{\text{easy}}(T)(m^{i}_{\text{easy}})^2 + \xi_{\text{hard}}(T)(m^{i}_{\text{hard}})^2}{m^{i}_{\text{easy}} + m^{i}_{\text{hard}}} \leq \xi_{\text{easy}}(T)(m^{i}_{\text{easy}} + m^{i}_{\text{hard}}).$$

This leads to

$$1 - \xi_{\text{easy}}(T) \leq \frac{m^{i+1}_{\text{easy}} + m^{i+1}_{\text{hard}}}{m^{i}_{\text{easy}} + m^{i}_{\text{hard}}} \leq 1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)},$$

and therefore

$$(1 - \xi_{\text{easy}}(T))^R \leq \frac{m^{R}_{\text{easy}} + m^{R}_{\text{hard}}}{m^{0}_{\text{easy}} + m^{0}_{\text{hard}}} \leq \left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}\right)^R.$$

Assume the number of queries, $T \times R$, is fixed. Then, the optimal T from the lower bound is

$$T^*_{\text{low}} = \arg\min_T \frac{1}{T} \log(1 - \xi_{\text{easy}}(T)).$$

By setting the derivative to be zero, we have T^*_{low} is the solution to

$$-T\xi'_{\text{easy}}(T) = (1 - \xi_{\text{easy}}(T)) \log(1 - \xi_{\text{easy}}(T)).$$

Similarly, the optimal T from the upper bound is

$$T^*_{\text{upp}} = \arg\min_T \frac{1}{T} \log\left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}\right).$$

By setting the derivative to be zero, we have T^*_{upp} is the solution to

$$-T.\frac{\xi'_{\text{easy}}(T)\xi_{\text{hard}}(T)^2 + \xi'_{\text{hard}}(T)\xi_{\text{easy}}(T)^2}{(\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T))^2} = \left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}\right) \log\left(1 - \frac{\xi_{\text{easy}}(T)\xi_{\text{hard}}(T)}{\xi_{\text{easy}}(T) + \xi_{\text{hard}}(T)}\right).$$

Appendix C
Feasibility of Discriminator in the Classifier-based Setting

The solution to (5) and (6) is:

$$D^*(x) = \begin{cases} \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1-\lambda)p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1-\lambda)p_\Omega} & \text{if } \mathbf{f}(x) \geq \tau \\ \alpha_- & \text{if } \mathbf{f}(x) < \tau \end{cases},$$

which satisfies $D^* \in [0, 1]$. Therefore, (5) is feasible with the `guide` function defined in (6).

APPENDIX D
EXPERIMENTAL SETUP

Pre-training. We use DCGAN [Radford et al., 2015] with latent dimension $= 128$ as the model. The pre-trained model is trained with label smoothing ($\alpha_+ = 0.9, \alpha_- = 0.1$):

$$\min_G \max_D \quad \mathbb{E}_{x\sim X}\left[\alpha_+ \log D(x) + (1-\alpha_+)\log(1-D(x))\right] \\ + \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\alpha_- \log D(G(z)) + (1-\alpha_-)\log(1-D(G(z)))\right].$$

We use Adam optimizer with learning rate $= 2\times 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.999$ to optimize both the generator and the discriminator. The networks are trained for 200 epochs with a batch size of 64. For each iteration over one mini-batch, we let K_D be the number of times to update the discriminator, and K_G the number of times to update the generator. We use $K_D = 1$ and $K_G = 5$ to train.

Data forgetting. The setup is similar to the pre-training except for two differences. The number of epochs is much smaller: 8 for MNIST, 30 for CIFAR, and 40 for STL-10. We let $K_G = 1$ for MNIST and CIFAR and $K_G = 5$ for STL-10.

Evaluation. To measure invalidity, we generate 50K samples, and compute the fraction of these samples that are not valid (e.g., classified as the label to be forgotten, or with pre-defined biases). It is the lower the better. The invalidity for forgetting labels is measured based on label classifiers. We use pre-trained classifiers on these datasets. [2]

The other evaluation metric is generation quality. The inception score (`IS`) [Salimans et al., 2016] is computed based on logit distributions from the above pre-trained classifiers. It is the higher the better. The Frechet Inception Distance (`FID`) [Heusel et al., 2017] is computed based on an open-sourced PyTorch implementation. [3] It is the lower the better.

When computing these quality metrics, we generate 50K samples, and compare to the set of valid training samples: $\{x \in X : x \not\in \Omega\}$. Therefore, when $X \cap \Omega$ is not the empty set (such as forgetting labels in Section IV-A), the quality measure of the model after data forgetting is not directly comparable to the pre-trained model, but these scores among different forgetting algorithms are comparable and give intuition to the generation quality. When $X \cap \Omega$ is the empty set (such as de-biasing in Section IV-B), the quality measures of the pre-trained model and the model after data forgetting are directly comparable.

[2] https://github.com/aaron-xichen/pytorch-playground (MIT license)
[3] https://github.com/mseitzer/pytorch-fid (Apache-2.0 license)

APPENDIX E
FORGETTING LABELS

A. Forgetting Label 0

1) Main results: We include results for forgetting label 0 in this section. We look at MNIST, CIFAR-10, and STL-10 datasets with different sets of hyper-parameters. With the base set of hyper-parameters, while different forgetting algorithms perform better on different datasets, they are highly comparable with each other. We find the results are worse when there is no label smoothing ($\alpha_+ = 1, \alpha_- = 0$), indicating label smoothing is important for data forgetting. We discuss results after one epoch in Appendix E-A2, the effect of λ in Appendix E-A4, and the effect of T in Table II.

Results for MNIST:

TABLE VII: Data-based forgetting algorithm.

Model	Epochs	Inv(↓)	IS(↑)
pre-trained	200	1.095×10^{-1}	7.82
Base: $\alpha_+ = 0.95, \alpha_- = 0.05, \lambda = 0.85$	8	0	7.19
$\alpha_+ = 0.9, \alpha_- = 0.1$	8	2×10^{-5}	7.02
$\alpha_+ = 1.0, \alpha_- = 0.0$	8	4×10^{-5}	6.97
$\lambda = 0.8$	8	2×10^{-5}	7.18
$\lambda = 0.9$	8	4×10^{-5}	7.16
$\lambda = 0.95$	8	5.2×10^{-4}	7.19
$\lambda = 0.8$	1	2.98×10^{-3}	7.09

TABLE VIII: Validity-based forgetting algorithm.

Model	Epochs	Inv(↓)	IS(↑)
pre-trained	200	1.095×10^{-1}	7.82
Base: $\alpha_+ = 0.95, \alpha_- = 0.05, \lambda = 0.85, T = 1000$	8	8×10^{-5}	7.17
$\alpha_+ = 0.9, \alpha_- = 0.1$	8	3.4×10^{-4}	7.06
$\alpha_+ = 1.0, \alpha_- = 0.0$	8	3.72×10^{-3}	4.81
$\lambda = 0.8$	8	0	7.23
$\lambda = 0.9$	8	2.2×10^{-4}	7.07
$\lambda = 0.95$	8	8.8×10^{-4}	7.12
$T = 400$	20	0	7.10
$T = 2000$	4	2.8×10^{-4}	7.11
$\lambda = 0.8$	1	2.80×10^{-3}	6.99

TABLE IX: Classifier-based forgetting algorithm.

Model	Epochs	Inv(↓)	IS(↑)
pre-trained	200	1.095×10^{-1}	7.82
Base: $\alpha_+ = 0.95, \alpha_- = 0.05, \lambda = 0.85, \tau = 0.5$	8	4×10^{-5}	7.19
$\alpha_+ = 0.9, \alpha_- = 0.1$	8	1.4×10^{-4}	7.09
$\alpha_+ = 1.0, \alpha_- = 0.0$	8	2.06×10^{-3}	6.08
$\lambda = 0.8$	8	6×10^{-5}	7.15
$\lambda = 0.9$	8	8×10^{-5}	7.18
$\lambda = 0.95$	8	7.2×10^{-4}	7.24
$\tau = 0.3$	8	1.2×10^{-4}	7.12
$\tau = 0.7$	8	6×10^{-5}	7.22
$\lambda = 0.8$	1	2.54×10^{-3}	7.11

Results for CIFAR-10:

TABLE X: Data-based forgetting algorithm.

Model	Epochs	Inv(↓)	FID(↓)
pre-trained	200	1.291×10^{-1}	36.2
Base: $\alpha_+ = 0.9, \alpha_- = 0.05, \lambda = 0.8$	30	7.4×10^{-3}	35.8
$\alpha_+ = 0.9, \alpha_- = 0.1$	30	8.0×10^{-3}	34.4
$\alpha_+ = 0.9, \alpha_- = 0.0$	30	8.9×10^{-3}	34.2
$\lambda = 0.9$	30	2.10×10^{-2}	29.2
$\lambda = 0.95$	30	4.21×10^{-2}	26.2
Base	1	3.99×10^{-2}	37.1

TABLE XI: Validity-based forgetting algorithm.

Model	Epochs	Inv(↓)	FID(↓)
pre-trained	200	1.291×10^{-1}	36.2
Base: $\alpha_+ = 0.9, \alpha_- = 0.05, \lambda = 0.8, T = 1000$	30	7.9×10^{-3}	35.3
$\alpha_+ = 0.9, \alpha_- = 0.1$	30	8.1×10^{-3}	33.8
$\alpha_+ = 0.9, \alpha_- = 0.0$	30	8.1×10^{-3}	34.1
$\lambda = 0.9$	30	2.54×10^{-2}	28.1
$\lambda = 0.95$	30	3.57×10^{-2}	27.8
$T = 400$	75	4.5×10^{-3}	35.1
$T = 2000$	15	1.00×10^{-2}	31.9
Base	1	3.85×10^{-2}	36.2

TABLE XII: Classifier-based forgetting algorithm.

Model	Epochs	Inv(↓)	FID(↓)
pre-trained	200	1.291×10^{-1}	36.2
Base: $\alpha_+ = 0.9, \alpha_- = 0.05, \lambda = 0.8, \tau = 0.5$	30	1.28×10^{-2}	33.7
$\alpha_+ = 0.9, \alpha_- = 0.1$	30	1.04×10^{-2}	32.9
$\alpha_+ = 0.9, \alpha_- = 0.0$	30	6.3×10^{-3}	32.8
$\lambda = 0.9$	30	2.25×10^{-2}	28.6
$\lambda = 0.95$	30	4.26×10^{-2}	26.9
$\tau = 0.3$	30	9.6×10^{-3}	34.8
$\tau = 0.7$	30	1.05×10^{-2}	35.2
Base	1	3.47×10^{-2}	37.8

Results for STL-10:

TABLE XIII: Data-based forgetting algorithm.

Model	Epochs	Inv(↓)	FID(↓)
pre-trained	200	6.23×10^{-2}	79.1
Base: $\alpha_+ = 0.9, \alpha_- = 0.05, \lambda = 0.8$	40	7.8×10^{-4}	74.3
$\alpha_+ = 0.9, \alpha_- = 0.1$	40	7.6×10^{-4}	75.8
$\alpha_+ = 0.9, \alpha_- = 0.0$	40	1.42×10^{-3}	82.7
$\lambda = 0.9$	40	2.88×10^{-3}	76.9
$\lambda = 0.95$	40	6.71×10^{-3}	78.2
Base	1	6.97×10^{-3}	75.1

TABLE XIV: Validity-based forgetting algorithm.

Model	Epochs	Inv(↓)	FID(↓)
pre-trained	200	6.23×10^{-2}	79.1
Base: $\alpha_+ = 0.9, \alpha_- = 0.05, \lambda = 0.8, T = 1000$	40	4.8×10^{-4}	79.3
$\alpha_+ = 0.9, \alpha_- = 0.1$	40	8.2×10^{-4}	76.5
$\alpha_+ = 0.9, \alpha_- = 0.0$	40	1.44×10^{-3}	77.0
$\lambda = 0.9$	40	4.52×10^{-3}	75.9
$\lambda = 0.95$	40	8.95×10^{-3}	75.3
$T = 400$	100	1.00×10^{-3}	75.1
$T = 2000$	20	1.00×10^{-3}	75.1
Base	1	8.99×10^{-3}	79.5

TABLE XV: Classifier-based forgetting algorithm.

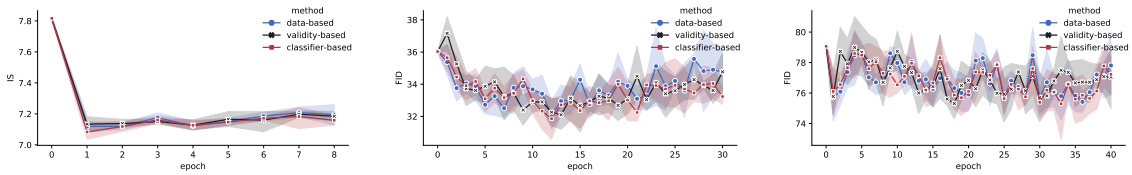
Model	Epochs	Inv(↓)	FID(↓)
pre-trained	200	6.23×10^{-2}	79.1
Base: $\alpha_+ = 0.9, \alpha_- = 0.05, \lambda = 0.8, \tau = 0.5$	40	8.6×10^{-4}	75.4
$\alpha_+ = 0.9, \alpha_- = 0.1$	40	9.2×10^{-4}	74.8
$\alpha_+ = 0.9, \alpha_- = 0.0$	40	1.62×10^{-3}	82.0
$\lambda = 0.9$	40	3.10×10^{-3}	77.2
$\lambda = 0.95$	40	6.89×10^{-3}	76.2
$\tau = 0.3$	40	8.8×10^{-4}	73.8
$\tau = 0.7$	40	1.34×10^{-3}	76.1
Base	1	6.81×10^{-3}	75.6

2) Invalidity after one epoch: We compare invalidity after only one epoch of data forgetting. These forgetting algorithms are highly comparable to each other. We hypothesis that the classifier-based algorithm performs the best on MNIST because a label classifier on MNIST (and its gradient information) can be very accurate, while this may not be true for CIFAR-10 and STL-10.

TABLE XVI: Invalidity after one epoch of data forgetting.

Dataset	Scale	Pre-trained	Data-based	Validity-based	Classifier-based
MNIST	$\times 10^{-3}$	1.1×10^{2}	4.7 ± 0.8	5.6 ± 0.9	$\mathbf{3.9 \pm 0.9}$
CIFAR-10	$\times 10^{-2}$	1.3×10^{1}	$\mathbf{3.7 \pm 0.5}$	3.7 ± 0.8	3.8 ± 0.3
STL-10	$\times 10^{-3}$	6.2×10^{1}	9.1 ± 0.9	$\mathbf{8.6 \pm 0.9}$	10.6 ± 1.2

3) Quality during data forgetting: We plot quality measure of different data forgetting algorithms on different datasets during the forgetting process, complementary to the invalidity in Fig. 2. We find the variances of quality measure is higher than the invalidity, but different forgetting algorithms are generally comparable.



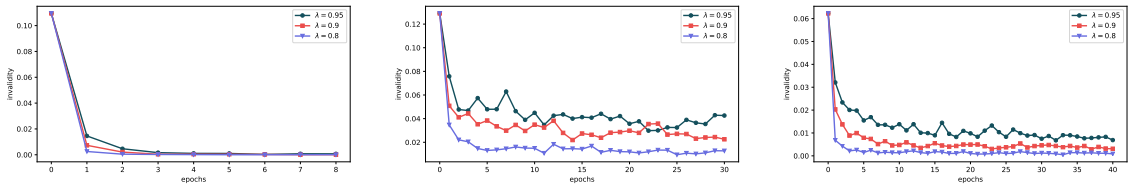
(a) MNIST (b) CIFAR-10 (c) STL-10

Fig. 5: Quality measure during data forgetting. Mean and standard errors are plotted for five random seeds.

4) Trade-off by alternating λ: We study the effect of λ (hyper-parameter in (2)) in Table XVII and Fig. 6. There is a trade-off by alternating λ: a larger λ (less fake data from the forgetting set) leads to better quality measure, and a smaller λ (more fake data from the forgetting set) leads to better invalidity.

TABLE XVII: Invalidity after data forgetting for different λ in the classifier-based forgetting algorithm.

λ	MNIST		CIFAR-10		STL-10	
	Inv(↓)	IS(↑)	Inv(↓)	FID(↓)	Inv(↓)	FID(↓)
0.8	0.6×10^{-4}	7.15	1.28×10^{-2}	33.7	0.86×10^{-3}	75.4
0.9	0.8×10^{-4}	7.18	2.25×10^{-2}	28.6	3.10×10^{-3}	77.2
0.95	7.2×10^{-4}	7.24	4.26×10^{-2}	26.9	6.89×10^{-3}	76.2



(a) MNIST (b) CIFAR-10 (c) STL-10

Fig. 6: Invalidity during data forgetting for different λ in the classifier-based forgetting algorithm.

B. Forgetting Other Labels

We also demonstrate results for forgetting other labels with our data forgetting algorithms. We use the base set of hyper-parameters in Appendix E-A. Similar to forgetting label 0, all forgetting algorithms can largely reduce invalidity, and they are highly comparable to each other. The classifier-based forgetting algorithm achieves slightly better generation quality on MNIST and CIFAR-10. In terms of different labels, we find some labels are harder to forget in the sense that the invalidity scores for these labels are higher than other scores, such as label 9 in MNIST, and label 3 in CIFAR-10 and STL-10.

TABLE XVIII: Forgetting other labels on MNIST.

Label	Pre-trained		Data-based		Validity-based		Classifier-based	
	Inv(↓)	IS(↑)	Inv(↓)	IS(↑)	Inv(↓)	IS(↑)	Inv(↓)	IS(↑)
1	10.2%	7.81	0.002%	7.01	**0.000%**	**7.21**	0.008%	7.13
2	8.6%	7.81	0.022%	7.22	**0.012%**	7.20	0.028%	**7.28**
3	11.5%	7.81	**0.126%**	7.20	0.136%	**7.24**	0.134%	7.19
4	9.9%	7.81	0.138%	7.19	**0.092%**	7.21	0.104%	**7.26**
5	8.7%	7.81	0.048%	7.22	**0.046%**	7.21	0.056%	**7.24**
6	9.0%	7.81	0.020%	7.04	0.022%	7.07	**0.010%**	**7.12**
7	11.4%	7.81	0.114%	7.24	0.124%	**7.34**	**0.088%**	7.32
8	9.1%	7.81	**0.198%**	7.48	0.248%	7.35	0.302%	**7.51**
9	10.7%	7.81	0.486%	7.30	**0.414%**	**7.36**	0.545%	7.26

TABLE XIX: Forgetting other labels on CIFAR-10.

Label	Pre-trained		Data-based		Validity-based		Classifier-based	
	Inv(↓)	FID(↓)	Inv(↓)	FID(↓)	Inv(↓)	FID(↓)	Inv(↓)	FID(↓)
1	1.5%	36.24	0.032%	35.06	**0.014%**	35.23	0.082%	**33.40**
2	11.0%	36.24	**1.311%**	31.67	1.537%	31.65	1.564%	**28.34**
3	15.8%	36.24	3.013%	30.10	3.491%	31.01	**2.534%**	**28.06**
4	16.8%	36.24	1.752%	30.36	1.754%	31.26	**1.590%**	**29.72**
5	6.7%	36.24	**0.799%**	**30.76**	0.985%	30.90	1.461%	31.36
6	9.3%	36.24	0.797%	29.81	1.071%	31.65	**0.755%**	**29.64**
7	8.6%	36.24	0.789%	33.48	**0.496%**	**33.40**	1.325%	34.15
8	10.3%	36.24	**0.218%**	38.96	1.451%	38.59	0.496%	**34.56**
9	7.1%	36.24	**0.138%**	38.13	0.186%	37.74	0.216%	**36.85**

TABLE XX: Forgetting other labels on STL-10.

Label	Pre-trained		Data-based		Validity-based		Classifier-based	
	Inv(↓)	FID(↓)	Inv(↓)	FID(↓)	Inv(↓)	FID(↓)	Inv(↓)	FID(↓)
1	9.0%	79.00	**1.273%**	74.89	2.168%	**73.91**	1.900%	75.34
2	6.2%	79.00	0.158%	**72.22**	**0.132%**	72.39	0.176%	75.75
3	14.9%	79.00	3.772%	77.24	**3.732%**	76.80	4.412%	**75.19**
4	8.2%	79.00	1.634%	**81.91**	**1.345%**	82.82	1.425%	83.25
5	15.1%	79.00	**2.072%**	**76.85**	3.383%	80.40	5.041%	77.74
6	8.7%	79.00	**0.462%**	80.82	0.518%	**78.17**	0.745%	79.63
7	10.7%	79.00	2.973%	**77.53**	**1.838%**	78.57	2.180%	77.58
8	9.5%	79.00	0.304%	79.56	**0.272%**	78.06	0.352%	**77.07**
9	11.6%	79.00	**0.817%**	76.70	0.947%	78.37	0.941%	**76.37**

C. Visualization

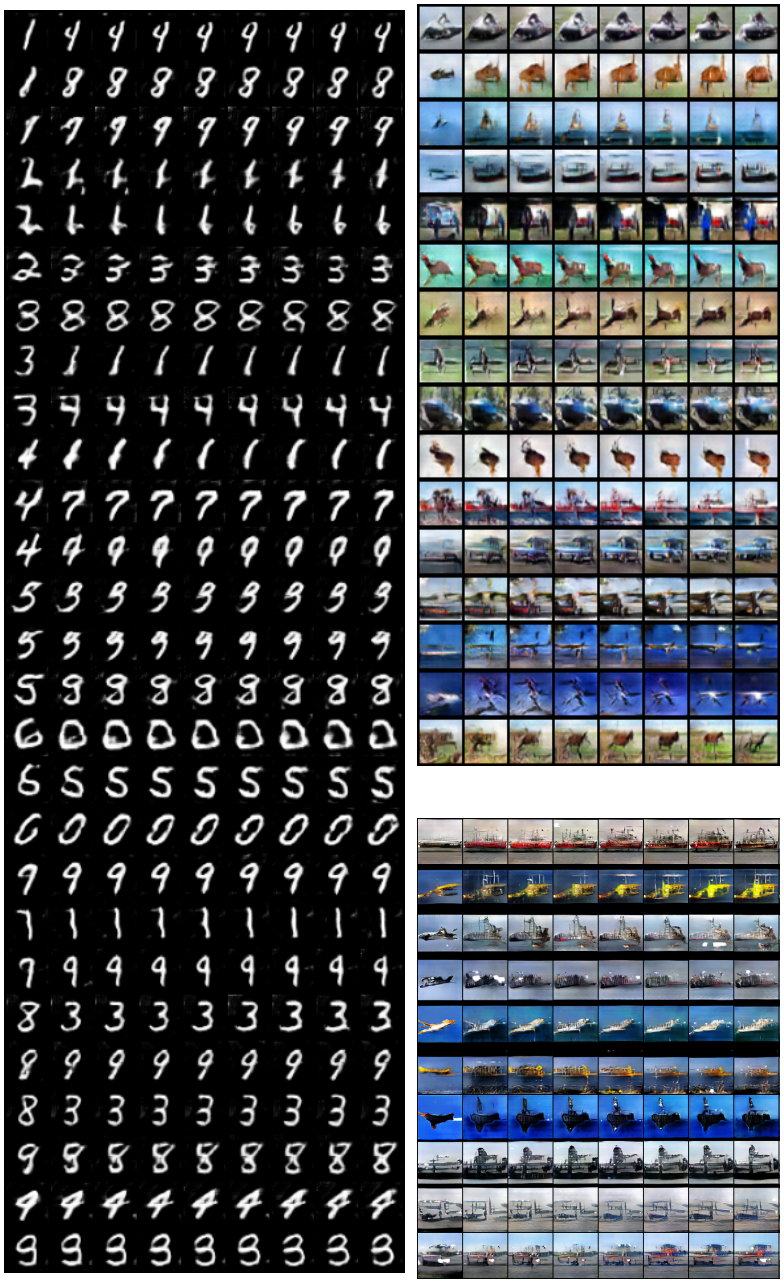


Fig. 7: Visualization of the data forgetting process of invalid samples when forgetting labels. The first column is generated by the pre-trained generator, and the i-th column is generated after $k \cdot (i-1)$ epochs of data forgetting. Left: MNIST with $k = 1$. Right: top is CIFAR-10 and bottom is STL-10, both with $k = 4$. We can see samples associated with invalid labels are gradually pushed to other labels, but a high-level visual similarity is kept.

D. Detailed Setup of Forgetting Multiple Sets

We use 30K images from CelebA-64 as the training set. All other hyper-parameters are the same as the base set for STL-10 in Appendix E-A, except that we run data forgetting algorithms for only 5 epochs. We train attribute classifiers for each attribute separately. The attribute classifiers are fine-tuned from open-sourced pre-trained ResNet [He et al., 2016]. [4] We fine-tune the network for 20 epochs using the SGD optimizer with learning rate $= 1 \times 10^{-3}$, momentum $= 0.9$, and a batch size of 64.

[4] https://pytorch.org/vision/stable/models.html

APPENDIX F
MODEL DE-BIASING

A. Boundary Artifacts

Let the image size be $W \times H$ (the number of channels is 1 for MNIST). For an integer margin, the boundary pixels are defined as

$$\{(i,j) : 1 \leq i \leq \text{margin or } W - \text{margin} < i \leq W, 1 \leq j \leq \text{margin or } H - \text{margin} < j \leq H\}.$$

Then, the validity function for boundary artifacts is defined as

$$\mathbf{v}(x) = 1 \left\{ \sum_{(i,j) \in \text{boundary pixels}} x_{ij} < \tau_b \right\},$$

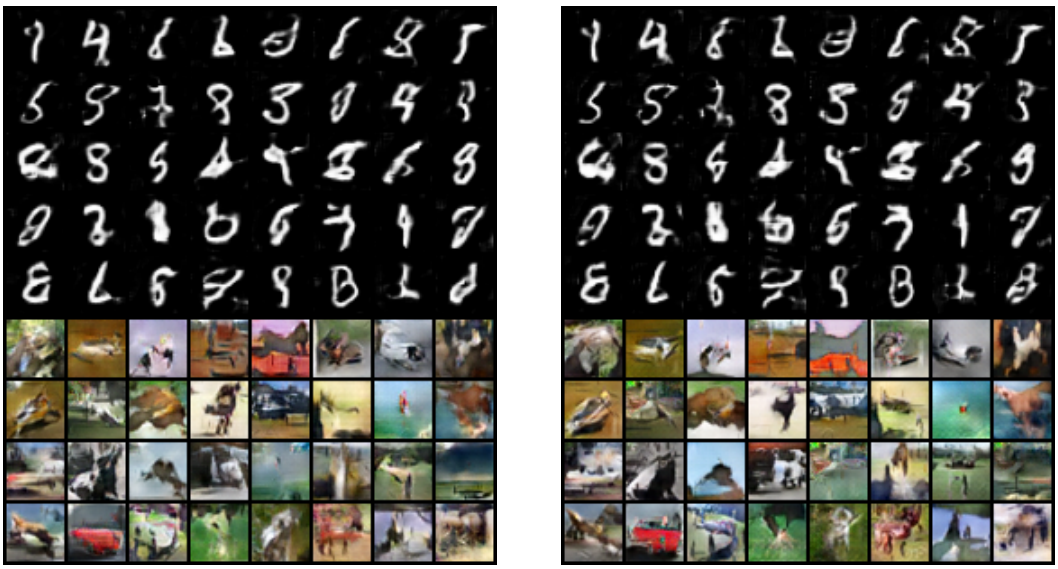
where $\tau_b = 4.25$ for $\text{margin} = 1$ and 10.0 for $\text{margin} = 2$. For these values, no training data has the boundary artifact. Quantitative results are in Tabel IV. We visualize some samples with boundary artifacts in Fig. 8a. We run the validity-based forgetting algorithm with $\lambda = 0.98, \alpha_+ = 0.95, \alpha_- = 0.05$ for 4 epochs. After de-biasing via data forgetting, these samples have less boundary pixels, as shown in Fig. 8b.

(a) Samples with boundary artifacts. (b) Samples after de-biasing via data forgetting.

Fig. 8: De-biasing boundary artifacts with the validity-based data forgetting algorithm. Margin $= 1$ and $T = 40$K.

B. Label Biases

We use classifier-based forgetting algorithm to de-bias label biases. For MNIST, we use $\lambda = 0.8, \alpha_+ = 0.95, \alpha_- = 0.05$ and run for 8 epochs. For CIFAR-10, we use $\lambda = 0.9, \alpha_+ = 0.9, \alpha_- = 0.05$ and run for 30 epochs. Quantitative results are in Table V and VI. We visualize semantically ambiguous samples generated by the pre-trained model in Fig. 9a. After de-biasing via data forgetting, these samples become less semantically ambiguous, as shown in Fig. 9b.



(a) Samples with label-biases.

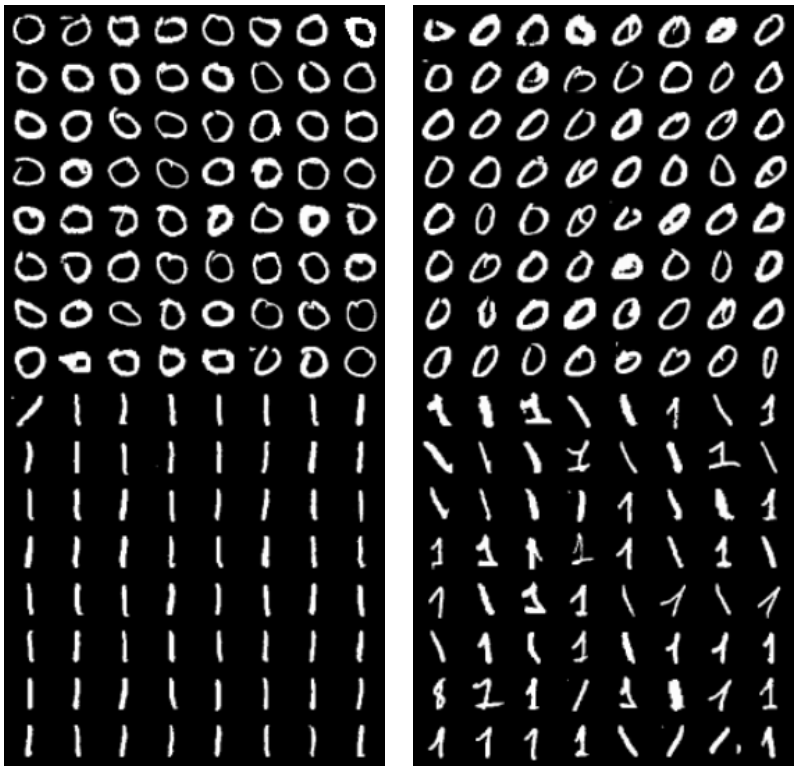
(b) Samples after de-biasing via data forgetting.

Fig. 9: De-biasing label biases with the classifier-based data forgetting algorithm ($\tau = 0.7$).

APPENDIX G
UNDERSTANDING TRAINING DATA

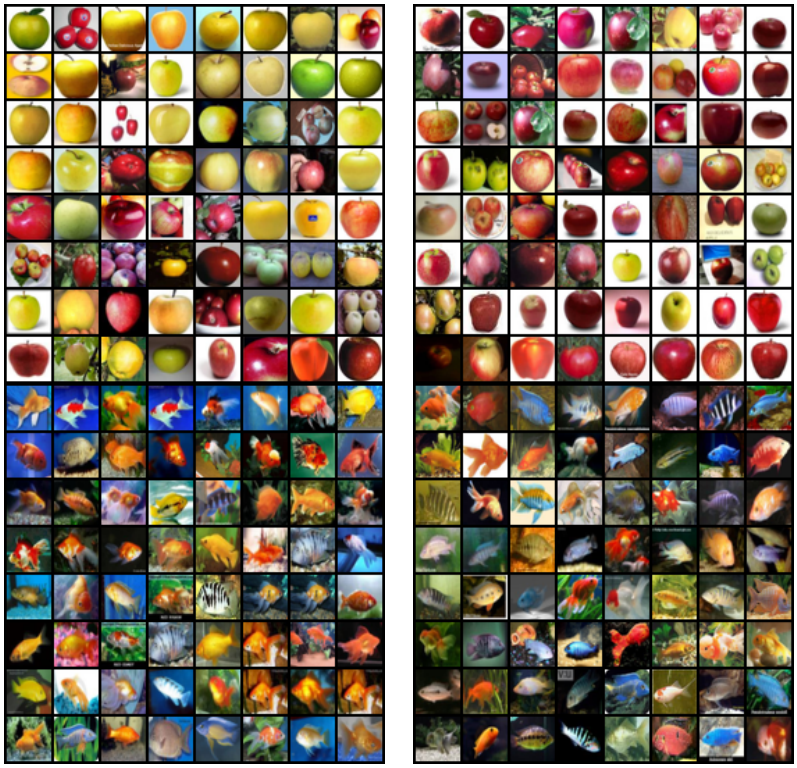
A. *Sample-level forgetting difficulty*

We visualize some most and least difficult-to-forget samples according to the forgetting scores in Fig. 10 and Fig. 11. We find the most difficult-to-forget samples are visually atypical, while the least difficult-to-forget samples are visually more common.



(a) Samples that are easiest to forget. (b) Samples that are hardest to forget.

Fig. 10: Samples that are most and least difficult-to-forget in MNIST.



(a) Samples that are easiest to forget. (b) Samples that are hardest to forget.

Fig. 11: Samples that are most and least difficult-to-forget in CIFAR-100.

B. Label-level forgetting difficulty

We sort all labels according to their average forgetting scores. This tells us which labels are easier or harder to forget. The results for MNIST are in Fig. 12. Consistent with Table XVIII, label 9 is the most difficult label to forget. The most and least difficult-to-forget labels for CIFAR-100 are shown in Fig. 13a and 13b.

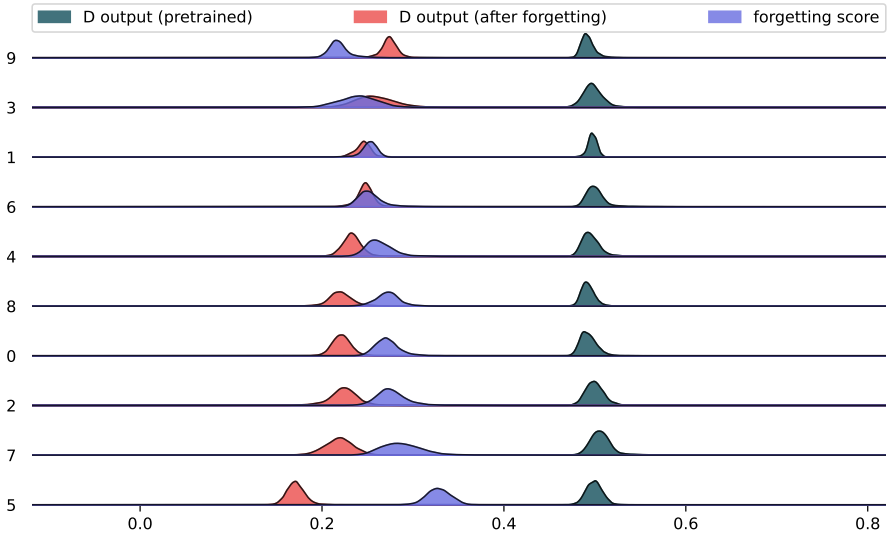
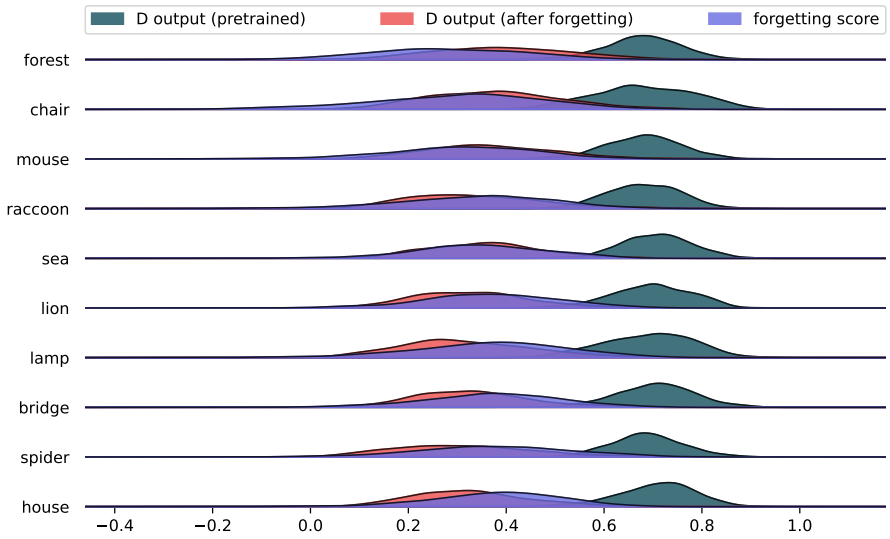
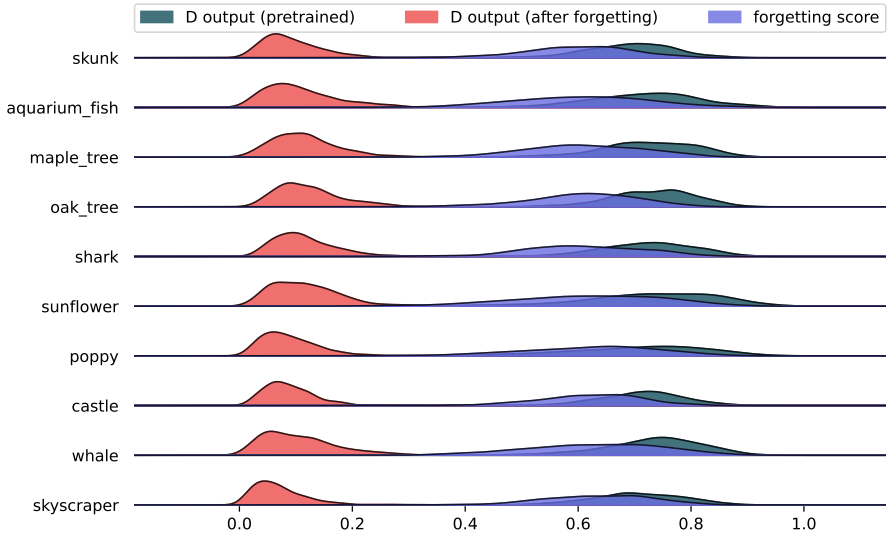


Fig. 12: Label-level forgetting difficulty for MNIST. Top: the most difficult to forget. Bottom: the least difficult to forget. A large forgetting score means a label is easier to be forgotten. We find some labels are more difficult to forget than others.



(a) Label-level forgetting difficulty for CIFAR-100 (10 most difficult-to-forget labels). Top: the most difficult to forget. Bottom: the least difficult to forget.



(b) Label-level forgetting difficulty for CIFAR-100 (10 least difficult-to-forget labels). Top: the most difficult to forget. Bottom: the least difficult to forget.

Fig. 13: Label-level forgetting difficulty for CIFAR-100. A large forgetting score means a label is easier to be forgotten. We find some labels are more difficult to forget than others.

APPENDIX H
RELATIONSHIP TO ADVERSARIAL SAMPLES

Consider the classifier-based forgetting algorithm. We fix the discriminator and only update the generator. Then, the generator is trained to fool both the discriminator and the classifier at the same time. We may define generated samples from this generator as on-manifold adversarial samples to $(D, \mathbf{f})$. Note that on-manifold samples are not necessarily visually clear samples; instead, they could be high likelihood samples according to the inductive bias of the generative model.

When training, we use the classifier-based forgetting algorithm with $\tau = 0.5, \alpha_+ = 0.95, \alpha_- = 0.05, \lambda = 0.85$, and a batch size of 64. We "forget" one label at a time, similar to experiments in Appendix E-B. After each iteration over one mini-batch, we generate samples with the same latents. The visualization is shown below. There are several interesting findings shown in the figures. First, the generated samples tend to have less pixels. Second, the generated samples tend to be dis-connected. Third, there are some general patterns across these generated samples (for each label): for example, there are pixels in the middle of zeroes, the bottom of sevens vanish, and nines are split from the middle. We conjecture that these observations correspond to the inductive bias of the discriminator and adversarial samples of the classifier.

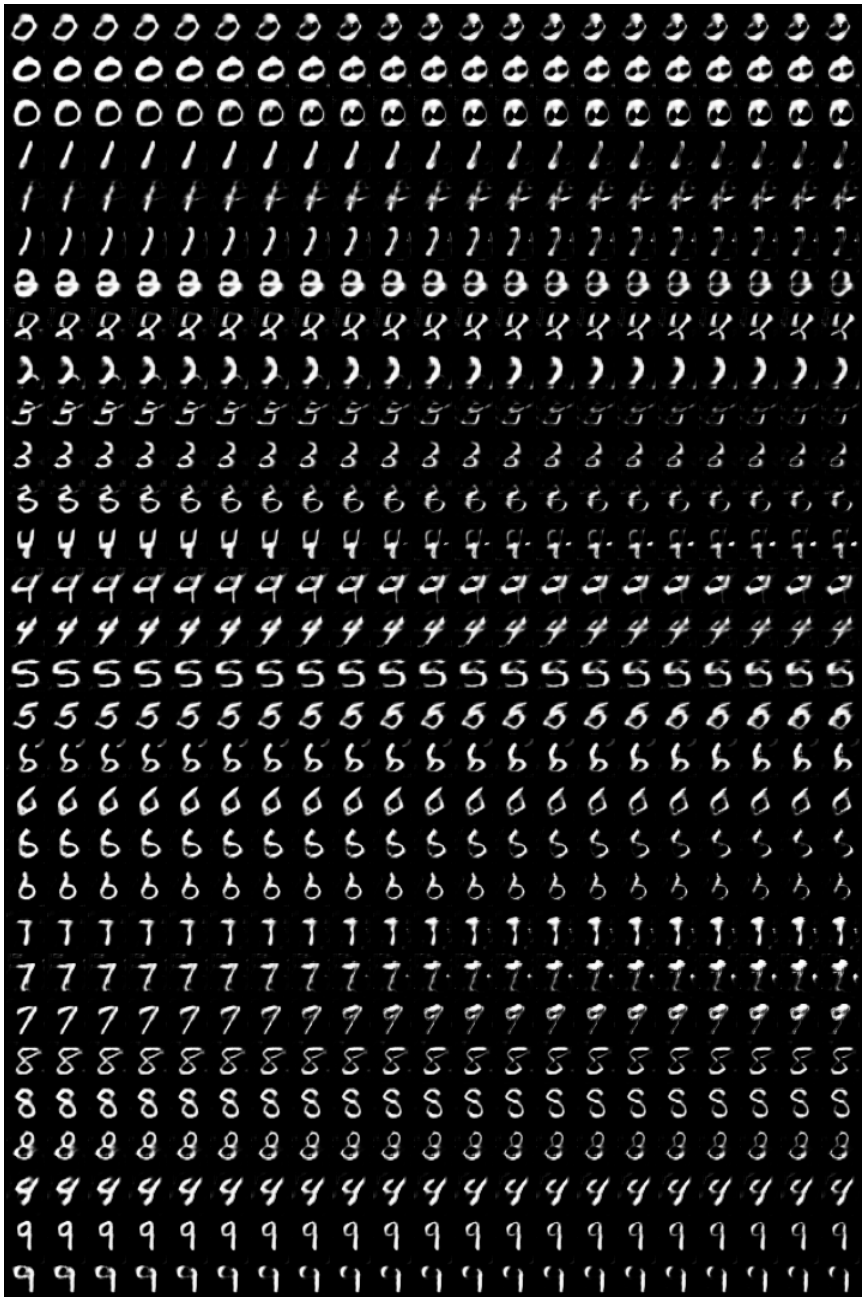


Fig. 14: Generated samples when we only train the generator and fix the discriminator and the classifier with the classifier-based forgetting algorithm. The first column is generated by the pre-trained generator, and the i-th column is generated after $i-1$ iterations (up to 20 iterations).

Reviews from Prior Submission

This paper was submitted to NeurIPS 2022 and withdrawn from NeurIPS 2022 before submitted to SaTML 2023. The SaTML 2023 version of our paper is a considerably revised version of the previously submitted one. We made three major improvements:

1. We better motivate the problem of post-editing a deep generative model in the introduction.
2. We make a distinction with data deletion in Section II-C.
3. We also added more data deletion baselines in the experiments.

We then attach the anonymized reviews, rebuttal, and the original submission below.

1 Official Review of Paper4614 by Reviewer vXyX

Summary

The paper proposes to modify GAN output distribution by retraining it to penalize certain samples, which can be constructed in various ways (direct samples, classifier, or score+threshold). In the new training steps, the fake distribution is a mixture of the output distribution and these samples, and the training procedure is identical to GAN training otherwise (with a few weighting and label smoothing tricks). On relatively simple datasets, the method can quite effectively suppress generation of a particular class, samples with visual borders, or even samples that are hard to classify. The results look visually convincing without much sacrifice of visual fidelity, which are verified via the quantitative scores The authors also propose a metric for measuring whether each sample is easy- or difficult- to forget, but however seems to be mostly affected by data likelihood/density (see questions below for a hypothesis why).

Strengths And Weaknesses

Strengths:

- The retraining algorithm is intuitive and effective.
- The presented forgetting results demonstrate very clean editing of the generator without much loss of visual quality.

Weakness:

- Unclear how to apply to non-GAN formulations (see question below).

- The authors claim potential debiasing application. But the method seems limited in this aspect. It is only able to completely remove certain outputs, rather than re-balance the generated distribution density.
- No results on more complex removal patterns: e.g., removal of all non-smile faces as done in [1].
- No comparison with other editing methods. While these methods may be proposed for a different purpose, it is still fair to compare with them. E.g., [1] considers rule-rewriting, which is arguably a superset of forgetting.

[1] D. Bau, S. Liu, T. Wang, J.-Y. Zhu, and A. Torralba. Rewriting a deep generative model. In European Conference on Computer Vision, pages 351–369. Springer, 2020.

Questions

Application beyond GANs

The method claims applicability on general generative models beyond GANs (line 116-117). But it is unclear to me how to do this when the method relies on access to a discriminator.

Debiasing

Is there a way to not complete remove certain samples, but simply reduce/increase their likelihoods? It would be much more useful for debiasing/fairness.

Complex removal patterns and comparison with prior works

Prior works investigate editing of quite complex patterns (e.g., facial expression, removing people, etc.) [1]. In principle the proposed method can also be applied to such tasks. Could the authors attempt such more complex forgetting sets? It would be good to know if the method still works well.

As mentioned above in weakness, while other editing methods may be proposed for a different purpose, it is still fair to compare with them. It would be good to compare against them on either the evaluated experiment settings, or some new more complex ones.

Forget score is related to data density? Uses of forget score?

Forget scores could be directly related to data density. E.g., if the generator is perfect, it starts with a discriminator score of 0. After forgetting, the optimal discriminator score is $\log p_G(x) - \log p_{\text{data}}(x)$, where x is the sample being forgotten. If generator forgets sufficiently well, $p_G(x)$ could be roughly equally small for different x. Then the score is dominated by the data density term. This could potentially explain the observation in Appendix G that atypical samples have lower scores. If this is true, then the forget score is essentially some measure of data density, and could potentially be useful (e.g., for pruning dataset size).

Cleaner method presentation.

I suggest merging the three method/algorithm explanations into one. They are essentially the same method, just with different ways to specify the removal set. It would be much cleaner to talk about only removal w.r.t. a set, and different ways to construct a set.

[1] D. Bau, S. Liu, T. Wang, J.-Y. Zhu, and A. Torralba. Rewriting a deep generative model. In European Conference on Computer Vision, pages 351–369. Springer, 2020.

Limitations

Generative modelling potentially have great social impact. Could the method be used by a malicious adversary for harms to society, e.g., by designing more biased generators? It would be good if the authors can discuss this, which are currently missing.

Ethics Flag: No
Soundness: 3 good
Presentation: 3 good
Contribution: 3 good
Rating: 4: Borderline reject: Technically solid paper where reasons to reject, e.g., limited evaluation, outweigh reasons to accept, e.g., good evaluation. Please use sparingly.
Confidence: 3: You are fairly confident in your assessment. It is possible that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work. Math/other details were not carefully checked.
Code Of Conduct: Yes

1.1 Response to Reviewer vXyX

Thanks for taking time to review our paper.

Q1: Application beyond GANs

In lines 116-117, we mean the different definitions of the forgetting set Ω applies to any generative model. The current forgetting algorithms are designed for GANs.

Q2: Debiasing

We think this is an excellent question. The current algorithm completely removes certain samples in theory. We believe that more complicated manipulations such as increasing or decreasing likelihood of certain samples can be achieved if the real and fake distributions in eq(3) are carefully designed for these tasks. We will investigate this as a future direction.

Q3: Complex removal patterns and comparison with prior works

We think our paper may not be directly comparable with Bau et al., 2020 in the sense that we are to learn a restricted distribution and avoid generating certain (potential undesirable) samples, while their work aims to edit generated samples (like a style transfer). However, we do think it is an excellent suggestion to consider complex patterns as in their work. We will investigate this in the future.

Q4: Forgetting scores

The forgetting score is defined as the difference between pre-trained and post-edited discriminator outputs. In figure 3 we show correlation between forgetting scores and pre-trained discriminator outputs, which might explain that atypical samples have lower scores.

As for "After forgetting, the optimal discriminator score is $\log p_G(x) - \log p_{\text{data}}(x)$": we do not see where this term comes from. We would appreciate it if you can explain more on this.

As for applications of forgetting scores such as data pruning: we think this is an interesting problem to look at. We believe that forgetting scores provide us a way to analyze and interpret samples from a dataset, and this can potentially facilitate many applications.

Q5: Presentation

The three algorithms differ in both (i) the description of the forgetting set, and (ii) intuitions behind the forgetting algorithms. The validity-based algorithm has an intuition of "using active learning as a backbone", and the classifier-based algorithm uses the gradient information of the classifier to achieve better and faster forgetting. Therefore, we think it is better to keep three separate algorithms.

Social impact

Our data forgetting algorithms are used for avoiding generating certain samples. Therefore, it is unlikely that an adversary can add bias into the model. We will add a discussion on this to the paper.

2 Official Review of Paper4614 by Reviewer u3qb

Summary

This paper focuses on learning good decision boundary between the real data and the fake data. Then authors explore this question from three perspectives: data-based, validation-based and classifier-based. Authors also provide the theoretical support. The quantitative and qualitative results demonstrate the effectiveness of the proposed method.

Strengths And Weaknesses

1. Authors explore training GANs from decision boundary by varying kinds of methods, which could improve GAN performance.
2. The paper is well-written,which helps readers easy to follow.
3. Authors valid the proposed methods on varying datasets and GAN variants.
4. The figure 1 looks clear and enjoyable.

Questions

I have a few questions:

1. The proposed methods are not varying interesting. For both data-based and validation-based methods, they lack of novelty, since current work already explored them. In fact authors used them, and improve a little bit, which is not enough for me.
2. The paper ensemble three methods to address the GANs training, which lacks of the core contribution and looks journey paper. The proposed three methods looks isolated, and does not have strong connection each other. I would like to see the method with deep exploration instead of different views.
3. Both validation-based and classifier -based method rely on validity function and differentiable classifier, which leads to more requirement.
4. The experiment is not enough for supporting the proposed method, since it only perform few comparison with few baselines. I suggest authors consider classical methods (e.g., sngan, sagan,pgan, stylegan,biggan). Also considering large dataset is interesting.

Limitations

Introducing three methods, which looks independent, makes readers believe that this paper lacks of the core contribution.

More baselines and datasets considered should be added.

Ethics Flag: No
Ethics Review Area: I don't know
Soundness: 1 poor
Presentation: 2 fair
Contribution: 1 poor
Rating: 3: Reject: For instance, a paper with technical flaws, weak evaluation, inadequate reproducibility and incompletely addressed ethical considerations.
Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.
Code Of Conduct: Yes

2.1 Response to Reviewer u3qb

Thanks for taking time to review our paper.

Q1: We believe the proposed methods have the following novelty:

(1) Data-based: we prove Theorem 1 (see proof in Appendix A), which provides a theoretical guarantee that the proposed algorithm will learn the restricted distribution. The theorem also applies to training GANs with label smoothing, a key contribution compared to Sinha et al., 2020. In addition, this algorithm is used to post-edit a pre-trained model, which is another novelty.

(2) Validity-based: the algorithm in Hanneke et al., 2018 only applies to simple and discrete distributions, with the forgetting algorithm being assumed as an (unknown) oracle. Our validity-based algorithm in the first practical instantiation and generalization of their algorithm with a clearly defined forgetting algorithm for complicated continuous high-dim distributions.

(3) Classifier-based: it is novel as we introduced this algorithm.

Q2: In this paper we consider three kinds of practical ways to describe forgetting sets. Based on the information we have for each description, we introduce different forgetting algorithms for each case. We think the connection between the three algorithms lies in this direction.

Q3: The use of forgetting algorithms depends on tasks, or more specifically, what information can be acquired. If there is a validity or classification function, then we can use the corresponding validity-based or classifier-based algorithms. If there are no such functions, then we use the data-based algorithm. In Table 1 we show these three algorithms have similar results.

Q4: Due to computation limits (we have access to one GPU), it could be challenging for us to investigate larger datasets. However, we think it is an excellent suggestion to investigate other types of GAN models. Space permitting, we will compare different GAN models and see if the results are general across different GANs.

3 Official Review of Paper4614 by Reviewer VLxc

Summary

The paper presents a novel approach for pre-trained GAN editing: removing some undesirable information (related with privacy concerns, ethics or for quality reasons). The authors present actually three methods for data removal (or forgetting): data-based (where a pre-specified set is provided), validity-based (where a validity check is performed), and classifier-based (where there is a differentiable classifier). The proposed approach has been tested on several datasets, demonstrating it is able to retain high quality generation capability at a fraction of the cost of full re-training.

Strengths And Weaknesses

Strengths:

1. the paper presents an interesting idea
2. the review of the state of the art is appropriate and covers most of the related works

Weaknesses:

1. the structure of the paper could be improved
2. the writing requires a significant improvement in clarity
3. despite your claims in the abstract that your method is tested on real-world datasets, actually you use standard datasets such as MNIST, CIFAR10, etc.
4. the experimental results are limited, and not convincing: there are no comparisons with other similar methods

Questions:

Here are my concerns:

1. Line 138: what is the meaning of the '#' symbol?
2. Eq. (3) and (4): G is not updated?
3. Regarding Method 1: it is not clear to me how do you prevent some samples not to be generated? Because I have the feeling you are talking here about forgetting at instance-level. You claim: "Omega could be any set of carefully designed or selected samples...". Please clarify this aspect.
4. Which criteria do you use to include samples in Omega? Despite that you claim that some samples will be 'deleted', what guarantee do you have that there will not be generated some other very similar ones? From the experimental results in table 1, I understand that you delete the whole class. So, are you deleting instances of a class or the whole class? It is not very clear for me. I would like to see a discussion on this topic.

5. Table 1 caption is confusing: "Invalidity and generation quality of different forgetting algorithms on forgetting label zero within different datasets". Label 0 applies only to MNIST. The other datasets have different labels. Please correct this aspect.

6. Section 3.2. The example regarding M (code generation model) and v (compiler) are totally unrelated with the context of this paper. Please provide some more appropriate examples and show how are they applied.

7. Method 2: your claim: "... this active algorithm focuses on invalid samples that are more likely to be generated, and therefore efficiently penalizes generation of invalid samples." So, this is more likely to be a look-ahead strategy? If I understood correctly, it is not a forgetting mechanism what are you implementing here, is just preventing that some samples to be generated? Please clarify this aspect.

8. Method 3: One of the criteria you mention is 'image inconsistency'. How did you measure the inconsistency of an image? So, the generated image is passed through a classifier which decides if the image has to be deleted or not?

9. You say in the abstract that your approach retains high quality generation capability at a fraction of the cost of full re-training. I haven't seen any experiment regarding this cost analysis.

10. The experimental results overall are limited. No comparison with other methods is provided.

Some minor comments (regarding paper structure):

1. Related work (subsection 1.1) should be a separate section (Section 2)

2. Section 2 could be moved as a sub-section of Section 3 (sub-section 3.1)

Limitations

Yes

Ethics Flag: No
Soundness: 2 fair
Presentation: 2 fair
Contribution: 2 fair
Rating: 3: Reject: For instance, a paper with technical flaws, weak evaluation, inadequate reproducibility and incompletely addressed ethical considerations.
Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.
Code Of Conduct: Yes

3.1 Response to Reviewer VLxc

Thanks for taking time to review our paper.

Regarding line 138

It means the push-forward distribution, i.e., the distribution defined by $G(z)$ where z is a Gaussian random variable.

Regarding eq (3) and (4)

G is updated according to line 143.

Regarding method 1

We prevent generating certain samples by defining the fake distribution in eq (2). Theorem 1 guarantees that the samples will not be generated. You are right in the sense that the forgetting is performed at instance-level. By saying "Ω can be any set ...", we mean it can be flexibly defined by the user, without any nontrivial constraint.

Regarding criteria to include samples in Ω

For the methodology, we assume the user defines Ω based on the specific task. Basically they can define any Ω and use our algorithms. The theoretical guarantee is in Theorem 1. As for experiments in Table 1, we define Ω as the whole class to evaluate how our algorithms perform on this task.

Regarding section 3.2

We think this example is very related to the context in the sense that in this case we do not want the model to generate incorrect (invalid) samples. As another example, the validity function can be an aggressive text filter in a language generative model.

Regarding method 2

It is related to active learning (as its backbone algorithm is an active learning algorithm). The algorithm actively finds where the generative model produces invalid samples, and then updates the mode to generate less such samples. Yes, it is preventing these samples from being generated, which is exactly what we call "forgetting".

Regarding method 3

Inconsistency is measured by the output logits of a classifier. If the output logits (after softmax) have a peak (like $(0.99, 0.01, 0, \cdots, 0)$), then it is a clear (consistent) sample. Otherwise, it is an inconsistent sample. This is formally defined in line 286 based on entropy of output logits. Yes, the generated image is passed through a classifier to decide whether it should be forgotten; our algorithm 3 is designed for this scenario.

Regarding cost analysis

Thanks for your reminder. We will add a formal discussion on this. To provide a qualitative analysis, we train GAN for 200 epochs in each experiment. In Table 1, we run our algorithms for a few epochs (8 for MNIST, 30 for CIFAR, and 40 for STL). As an example, for the data-based algorithm on MNIST, 1 step of forgetting (1 discriminator update and 1 generator update) takes less computation resources as 1 step of training (1 discriminator update and 5 generator updates). Therefore, the forgetting algorithm takes $< 4\%$ of computation as pre-training. The computation costs for validity or classifier based algorithms can be a little higher due to the extra operations, but they are roughly the same as the data-based algorithm.

Regarding baseline

We define the notion of data-forgetting in this paper. We believe that currently there are no other baseline methods for this task.

Forgetting Data from Pre-trained GANs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large pre-trained generative models are known to occasionally provide samples that may be undesirable for various reasons. The standard way to mitigate this is to re-train the models differently. In this work, we take a different, more compute-friendly approach and investigate how to post-edit a model after training so that it forgets certain kinds of samples. We provide three different algorithms for GANs that differ on how the samples to be forgotten are described. Extensive evaluations on real-world image datasets show that our algorithms are capable of forgetting data while retaining high generation quality at a fraction of the cost of full re-training.

1 Introduction

Generative adversarial networks (GANs) are a class of deep generative models that aim to learn a complicated probability distribution from data and generate samples from it [Goodfellow et al., 2014a]. These models have been immensely successful in many large scale tasks from multiple domains [de Masson d'Autume et al., 2019, Tulyakov et al., 2018, Karras et al., 2020, Kong et al., 2020, Zhu et al., 2020, Zhang et al., 2021]. However, many generative models may suffer from undesirable artifacts after training. For example, image models may generate blurred samples [Kaneko and Harada, 2021] or checkerboard artifacts [Odena et al., 2016, Zhang et al., 2019, Wang et al., 2020, Schwarz et al., 2021], speech models may produce unnatural sound [Donahue et al., 2018, Thiem et al., 2020], and language models may emit offensive text [Abid et al., 2021, Perez et al., 2022]. An important question is how to mitigate these artifacts.

Prior work has looked into mitigation by re-designing the pipeline including data augmentation, model architecture, loss functions, and then re-training the model [Isola et al., 2017, Aitken et al., 2017, Kaneko and Harada, 2021]. However, this may involve a lot of computation resources as modern GANs are extremely large and expensive to train. In addition, problems may become apparent after training, and resolving them may require multiple re-trainings. To address this challenge, we consider *post-editing*, which means modifying a pre-trained model in a certain way rather than training from scratch. This is a much more efficient process. It has shown empirical success in many supervised learning tasks [Frankle and Carbin, 2018, Zhou et al., 2021, Taha et al., 2021], but has not been studied as much for unsupervised learning. In particular, we propose a post-editing framework to forget undesirable samples that might be generated by a GAN, which we call *data forgetting*.

A naive solution to removing undesirable samples is using a classifier to filter them after generation. This approach, however, has several drawbacks. First, classifiers can take a significant amount of space and time after deployment, especially when there are multiple classifiers. Second, in cases where we provide a generative model to a third party, we have no control over whether they will use the filter. Finally, in the online or continual learning paradigm, we may combine data forgetting and fitting new samples in order to learn a dynamic distribution [Zhou et al., 2021]. If we use classifiers, a sample rejected at the beginning cannot be learned later. Data forgetting via post-editing, on the other hand, poses a cleaner solution which does not suffer from these limitations.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

There are two challenges in data forgetting. The first is how to describe the samples to be forgotten. This is important as data forgetting algorithms need to be tailored to specific descriptions. The second challenge is that we need to carefully balance between data forgetting and keeping good generation quality, which means the latent space and the networks must be carefully manipulated.

In this work, we propose a systematic framework for forgetting data from pre-trained generative models (see Section 2). We model data forgetting as learning the data distribution restricted to the complement of a forgetting set Ω. We then formalize three ways of describing forgetting sets, namely data-based (where a pre-specified set is given), validity-based (where there is a validity checker), and classifier-based (where there is a differentiable classifier).

Then, we introduce three data forgetting algorithms, one for each description (see Section 3). Prior works have looked at avoiding negative samples in the re-training setting with different descriptions and purposes [Sinha et al., 2020, Asokan and Seelamantula, 2020]. They introduce fake distributions to penalize the generation of negative samples. We extend this idea to data forgetting by defining the fake distribution as a mixture of the generative distribution and a forgetting distribution supported on Ω. We then prove the optimal generator can recover the target distribution when label smoothing [Salimans et al., 2016, Szegedy et al., 2016, Warde-Farley and Goodfellow, 2016] is used. Based on the theory, we introduce the data-based forgetting algorithm (Alg. 1). We then combine this algorithm with an improper active learning algorithm by Hanneke et al. [2018] and introduce the validity-based forgetting algorithm (Alg. 2). Finally, we propose to use a `guide` function to guide the discriminator via a classifier, and introduce the classifier-based forgetting algorithm (Alg. 3).

We evaluate these forgetting algorithms via experiments on real-world image datasets (see Section 4). We first show these algorithms can forget quickly while keeping high generation quality. We find they are generally comparable. We then investigate applications of data forgetting. We use these algorithms to remove different biases that may not exist in the training set but are learned by the pre-trained model. By data forgetting we can reduce such biases and improve generation quality. We also use data forgetting to understand training data. We investigate which samples are easy or hard to forget through forgetting scores. We finally discuss a potential definition of on-manifold adversarial samples via a variant of the classifier-based forgetting algorithm.

1.1 Related Work

Generative adversarial networks (GANs) [Goodfellow et al., 2014a] have achieved great success in a wide range of challenging, large-scale tasks. Examples include image generation [Karras et al., 2020], point cloud completion [Zhang et al., 2021], speech synthesis [Kong et al., 2020], video generation [Tulyakov et al., 2018], and language models [de Masson d'Autume et al., 2019].

Generative models may have different types of artifacts, such as blurred samples [Kaneko and Harada, 2021], fairness issues [Tan et al., 2020, Karakas et al., 2022], and checkerboard artifacts [Odena et al., 2016, Zhang et al., 2019, Wang et al., 2020, Schwarz et al., 2021] in image generation, offensive text in language models [Abid et al., 2021, Perez et al., 2022], and unnatural sound in speech models [Donahue et al., 2018, Thiem et al., 2020]. Many papers use post-editing to remove artifacts and improve GANs. Examples include improving fairness [Tan et al., 2020, Karakas et al., 2022], rule rewriting [Bau et al., 2020], discovering interpretability [Härkönen et al., 2020], and fine-tuning [Mo et al., 2020, Li et al., 2020, Zhao et al., 2020]. The purposes, use cases, and editing methods of these papers are different from data forgetting in our paper.

The most related papers to our paper are NDA [Sinha et al., 2020], Rumi-GAN [Asokan and Seelamantula, 2020], and Hanneke et al. [2018]. The first two papers look at how to avoid generating negative samples in the re-training setting. This is done by defining new fake distributions to penalize the generation of these samples. However, their purposes are different from us: NDA is used to characterize the boundary of the support of the generative distribution more precisely, and Rumi-GAN is used to handle unbalanced data. We extend their idea and theory to data forgetting in Section 3. [1] Hanneke et al. [2018] propose an active learning approach to avoid generating invalid samples, also in the re-training setting. However, they only carry theoretical study for discrete distributions. In our paper, the validity-based forgetting algorithm (Alg. 2) is based on a simplified version of their algorithm. We also use their definition of *invalidity* as an evaluation method.

[1] The loss functions in NDA and Rumi-GAN are similar.

There is a related concept called data deletion or machine unlearning [Cao and Yang, 2015]. It is different from data forgetting because these methods aim to approximate the re-trained model when some training samples are removed – mostly due to privacy reasons – while in data forgetting we penalize the model from knowing samples that should be forgotten. In addition, most data deletion techniques are for supervised learning or clustering, and is much less studied for generative models.

There is also a line of work on catastrophic forgetting in supervised learning [Kirkpatrick et al., 2017] and generative models [Thanh-Tung and Tran, 2020]. This concept is different from data forgetting in that we would like the generative model to forget certain data after training, while catastrophic forgetting means knowledge learned in previous tasks is destroyed during continual learning.

2 A Formal Framework for Data Forgetting

Let p_{data} be the data distribution on $\mathbb{R}^d$ and $X \sim p_{\text{data}}$ be i.i.d. training samples. Let $\mathcal{A}$ be a generative model and $\mathcal{M} = \mathcal{A}(X)$ be the pre-trained model on X, which learns p_{data}. In this paper, we consider $\mathcal{A}$ to be a GAN learning algorithm [Goodfellow et al., 2014a], and $\mathcal{M}$ contains two networks, D (discriminator) and G (generator), which are jointly trained to optimize

$$\min_G \max_D \ \mathbb{E}_{x\sim p_{\text{data}}} \log D(x) + \mathbb{E}_{z\sim\mathcal{N}(0,I)} \log(1 - D(G(z))). \tag{1}$$

Let the forgetting set $\Omega \subset \mathbb{R}^d$ be the set of samples we would like the model to forget. Formally, the goal is to develop a forgetting algorithm $\mathcal{D}$ such that $\mathcal{M}' = \mathcal{D}(\mathcal{M}, \Omega)$ learns the data distribution restricted on the complement $\bar{\Omega} = \mathbb{R}^d \setminus \Omega$, i.e. $p_{\text{data}}|_{\bar{\Omega}}$. Examples of Ω include inconsistent, blurred, unrealistic, or banned samples that are possibly generated by the model.

The forgetting set Ω, in addition to the pre-trained model, is considered as an input to the forgetting algorithm. We consider three kinds of Ω, namely data-based, validity-based, and classifier-based. First, the data-based Ω is a pre-defined set of samples in $\mathbb{R}^d$, such as a transformation applied on all training samples [Sinha et al., 2020]. Second, the validity-based Ω is defined as all invalid samples according to a validity function $\mathbf{v} : \mathbb{R}^d \to \{0, 1\}$, where 0 means invalid and 1 means valid. This is similar to the setting in Hanneke et al. [2018]. Finally, let $\mathbf{f} : \mathbb{R}^d \to [0, 1]$ be a soft classifier that outputs the probability that a sample belongs to a certain binary class, and $\tau \in (0, 1)$ be a threshold. Then, the classifier-based Ω is defined as $\{x : \mathbf{f}(x) < \tau\}$. For example, $\mathbf{f}$ can be an offensive text classifier in language generation tasks [Pitsilis et al., 2018]. Our framework is general and applies to any kind of generative model besides GANs.

The way data forgetting differs from data deletion is that in data forgetting, we want the model to assign zero likelihood to the forgetting set Ω in order to avoid generating samples from this region; however, in data deletion, there is a set $X' \subset X$ to be deleted, and the goal is to approximate the re-trained model $\mathcal{A}(X \setminus X')$.

3 Methods

In this section, we describe algorithms for each kind of forgetting set described in Section 2. We also provide theory on the optimality of the generator and the discriminator. Finally, we generalize the algorithms to situations where we would like the model to forget the union of multiple forgetting sets.

3.1 Data-based Forgetting Set

The data-based forgetting set Ω is a pre-defined set of samples we would like the model to forget. One example is a transformation function `NegAug` applied to all training samples, where `NegAug` makes realistic images unrealistic or inconsistent [Sinha et al., 2020]. Another example can be visually nice samples outside data manifold when the training set is small [Asokan and Seelamantula, 2020].

In our framework, the forgetting set Ω can be any set of carefully designed or selected samples depending on the purpose of forgetting them – which includes but does not limit to improving the generation quality of the model. For example, we expect the model to improve on fairness, bias, ethics or privacy when Ω is properly constructed with unfair, biased, unethical, or atypical samples.

To forget Ω, we regard both generated samples and Ω to be fake samples, and all training samples that are not in Ω to be real samples [Sinha et al., 2020, Asokan and Seelamantula, 2020]. Let p_Ω be

Algorithm 1 Forgetting Algorithm for Data-based Forgetting Set

Inputs: Pre-trained model $\mathcal{M} = (G_0, D_0)$, train set X, forgetting set Ω.
Initialize $G = G_0$, $D = D_0$.
Define the fake data distribution p_{fake} according to (2) with $p_\Omega = \mathcal{U}(\Omega)$.
Train G, D to optimize (3): $\min_G \max_D L(G, D)$.
return $\mathcal{M}' = (G, D)$.

Algorithm 2 Forgetting Algorithm for Validity-based Forgetting Set

Inputs: Pre-trained model $\mathcal{M} = (G_0, D_0)$, train set X, validity function $\mathbf{v}$.
Initialize $\Omega' = \{x \in X : \mathbf{v}(x) = 0\}$, $\mathcal{M}_0 = \mathcal{M}$.
for $i = 0, \cdots, R-1$ **do**
  Initialize $G = G_i$, $D = D_i$. Draw T samples $X_{\text{gen}}^{(i)}$ from G_i.
  Query $\mathbf{v}$ and add invalid samples to Ω': $\Omega' \leftarrow \Omega' \cup \{x \in X_{\text{gen}}^{(i)} : \mathbf{v}(x) = 0\}$.
  Define the fake data distribution p_{fake} according to (2) with $p_\Omega = \mathcal{U}(\Omega')$.
  Let $\mathcal{M}_{i+1} = (G_{i+1}, D_{i+1})$ optimize (3): $\min_G \max_D L(G, D)$.
end for
return $\mathcal{M}' = (G_R, D_R)$

Algorithm 3 Forgetting Algorithm for Classifier-based Forgetting Set

Inputs: Pre-trained model $\mathcal{M} = (G_0, D_0)$, train set X, differentiable classifier $\mathbf{f}$.
Initialize $G = G_0$, $D = D_0$.
Define the fake data distribution p_{fake} according to (2) with $p_\Omega = \mathcal{U}(\{x \in X : \mathbf{f}(x) < \tau\})$.
Train G, D to optimize (3): $\min_G \max_D L(G, \texttt{guide}(D, \mathbf{f}))$, where $\texttt{guide}(\cdot, \cdot)$ is defined in (6).
return $\mathcal{M}' = (G, D)$.

a distribution such that $\mathbf{supp}(p_\Omega) = \Omega$. Then, the fake data distribution p_{fake} is a mixture of the generative distribution $p_G = G\#\mathcal{N}(0, I)$ and p_Ω:

$$p_{\text{fake}} = \lambda \cdot p_G + (1 - \lambda) \cdot p_\Omega, \tag{2}$$

where $\lambda \in (0, 1)$ is a hyperparameter. We also apply label smoothing [Salimans et al., 2016, Szegedy et al., 2016, Warde-Farley and Goodfellow, 2016] techniques to the minimax loss function in order to improve robustness. Let $\alpha_+ \in (\frac{1}{2}, 1]$ be the positive target (such as 0.9) and $\alpha_- \in [0, \frac{1}{2})$ be the negative target (such as 0.1). Then, the loss function is

$$\begin{aligned} L(G, D) = \quad & \mathbb{E}_{x \sim p_{\text{data}}|_{\bar{\Omega}}} \left[\alpha_+ \log D(x) + (1 - \alpha_+) \log(1 - D(x))\right] \\ & + \mathbb{E}_{x \sim p_{\text{fake}}} \left[\alpha_- \log D(x) + (1 - \alpha_-) \log(1 - D(x))\right]. \end{aligned} \tag{3}$$

Theorem 1. *The optimal solution to* $\min_G \max_D L(G, D)$ *is*

$$D^* = \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1 - \lambda) p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1 - \lambda) p_\Omega}; \quad p_{G^*} = p_{\text{data}}|_{\bar{\Omega}}. \tag{4}$$

We provide the proof and theoretical extension to the more general f-GAN [Nowozin et al., 2016] setting in Appendix A. In the data-based setting, we let $p_\Omega = \mathcal{U}(\Omega)$, the uniform distribution on Ω. We assume Ω has positive, finite Lebesgue measure in $\mathbb{R}^d$ so that $\mathcal{U}(\Omega)$ is well-defined. The proposed method is summarized in Alg. 1.

Our objective function is connected to Sinha et al. [2020] and Asokan and Seelamantula [2020] in the sense that p_Ω is an instance of the negative distribution described in their frameworks. However, there are several significant differences between our method and theirs: (1) we start from a pre-trained model, (2) we aim to learn $p_{\text{data}}|_{\bar{\Omega}}$ rather than p_{data} and therefore do not require $\Omega \cap \mathbf{supp}(p_{\text{data}})$ to be the empty set, and (3) we use label smoothing techniques to improve robustness and provide theory for this setting. These differences are also true in the following sections.

3.2 Validity-based Forgetting Set

Let $\mathbf{v} : \mathbb{R}^d \to \{0, 1\}$ be a validity function that indicates whether a sample is valid. Then, validity-based forgetting set Ω is the set of all invalid samples $\{x : \mathbf{v}(x) = 0\}$. For example, $\mathcal{M}$ is a

code generation model, and $\mathbf{v}$ is a compiler that indicates whether the code is free of syntax errors [Hanneke et al., 2018]. Different from the data-based setting, the validity-based Ω may have infinite Lebesgue measure, such as a halfspace, and consequently $\mathcal{U}(\Omega)$ may not be well-defined.

To forget Ω, we let p_Ω in (2) to be a mixture of $p_{\text{data}}|_\Omega$ and $p_G|_\Omega$. This corresponds to a simplified version of the improper active learning algorithm introduced by Hanneke et al. [2018] with our Alg. 1 as their optimization oracle. The idea is to apply Alg. 1 for R rounds. After each round, we query the validity of T newly generated samples and use invalid samples to form a data-based forgetting set Ω'. In contrast to the data-based approach, this active algorithm focuses on invalid samples that are more likely to be generated, and therefore efficiently penalizes generation of invalid samples. The proposed method is summarized in Alg. 2.

The total number of queries to the validity function $\mathbf{v}$ is $|X| + T \times R$. In case $\mathbf{v}$ is expensive to run, we would like to achieve better data forgetting within a limited number of queries. From the data-driven point of view, we hope to collect as many invalid samples as possible. This is done by setting $R = 1$ and T maximized if we assume less invalid samples are generated after each iteration. However, this may not be the case in practice. We hypothesis some samples are easier to forget while others harder. By setting $R > 1$, we expect an increasing fraction of invalid generated samples to be hard to forget after each iteration. Focusing on these hard samples can potentially help the generator forget them. Since it is hard to directly analyze neural networks, we leave the rigorous study to future work. In Appendix B, we study a much simplified dynamical system corresponding to Alg. 2, where we show the invalidity (the mass of p_G on Ω) converges to zero, and provide optimal T and R values.

3.3 Classifier-based Forgetting Set

We would like the model to forget samples with certain (potentially undesirable) property. Let $\mathbf{f} : \mathbb{R}^d \to [0, 1]$ be a soft binary classifier on the property (0 means having the property and 1 means not having it), and $\tau \in (0, 1)$ be a threshold. The classifier-based forgetting set Ω is then defined as $\{x : \mathbf{f}(x) < \tau\}$. For example, the property can be *being offensive* in language generation, *containing no speech* in speech synthesis, or *visual inconsistency* in image generation. We consider $\mathbf{f}$ to be a trained machine learning model that is fully accessible and differentiable.

To forget Ω, we let p_Ω be a mixture of $p_{\text{data}}|_\Omega$ and $p_G|_\Omega$, similar to the validity-based approach. We use $\mathbf{f}$ to guide the discriminator and make it able to easily detect samples from Ω. Let $\texttt{guide}(D, \mathbf{f})$ be a guided discriminator that assigns small values to x when $\mathbf{f}(x) < \tau$ or $D(x)$ is small (i.e. $x \sim p_{\text{fake}}$), and large values to x when $\mathbf{f}(x) > \tau$ and $D(x)$ is large (i.e. $x \sim p_{\text{data}}|_{\bar{\Omega}}$). Instead of optimizing $L(G, D)$ in (3), we optimize $L(G, \texttt{guide}(D, \mathbf{f}))$. This will effectively update G by preventing it from generating samples in Ω. According to **Theorem** 1, the optimal discriminator is the solution to

$$\texttt{guide}(D^*, \mathbf{f}) = \frac{\alpha_+ p_{\text{data}}|_{\bar{\Omega}} + \alpha_-(\lambda p_G + (1-\lambda)p_\Omega)}{p_{\text{data}}|_{\bar{\Omega}} + \lambda p_G + (1-\lambda)p_\Omega}. \tag{5}$$

Therefore, the design of the `guide` function must make (5) feasible. In this paper, we let

$$\texttt{guide}(D, \mathbf{f})(x) = D(x) \text{ if } \mathbf{f}(x) \geq \tau, \text{ otherwise } \alpha_- + (D(x) - \alpha_-)\mathbf{f}(x). \tag{6}$$

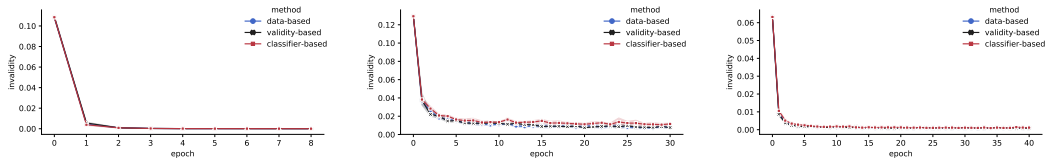
The feasibility of (5) is discussed in Appendix C. The proposed method is summarized in Alg. 3. The classifier-based Ω generalizes the validity-based Ω. First, any validity-based Ω can be represented by a classifier-based Ω if we let $\mathbf{f} = \mathbf{v}$ and $\tau = \frac{1}{2}$. Next, we note there is a trivial way to deal with classifier-based Ω via the validity-based approach – by setting $\mathbf{v}(x) = 1\{\mathbf{f}(x) < \tau\}$. However, potentially useful information such as values and gradients of $\mathbf{f}$ are lost, and we will evaluate this effect in experiments. In addition, the classifier-based approach does not maintain the potentially large set of invalid generated samples, as this step is automatically done in the `guide` function.

3.4 Generalization to Multiple Forgetting Sets

Let $\{\Omega_k\}_{k=1}^K$ be disjoint sets in $\mathbb{R}^d$, and we would like the model to forget $\Omega = \bigcup_{k=1}^K \Omega_k$. In the data-based setting, we let $p_\Omega = \mathcal{U}(\Omega) = \mathcal{U}(\bigcup_{k=1}^K \Omega_k)$. In the validity-based setting, each Ω_k is associated with a validity function $\mathbf{v}_k$. We let the overall validity function to be $\mathbf{v}(x) = \min_k \mathbf{v}_k(x)$. In the classifier-based setting, each Ω_k is associated with a classifier $\mathbf{f}_k$. Similar to the validity-based setting, we let the overall $\mathbf{f}$ to be $\mathbf{f}(x) = \min_k \mathbf{f}_k(x)$.

Table 1: Invalidity and generation quality of different forgetting algorithms on forgetting label zero within different datasets. Mean and standard errors are reported for five random seeds. Note that quality measure after data forgetting is not directly comparable with the pre-trained model. The invalidity drops in magnitude after data forgetting. Different forgetting algorithms are highly comparable to each other.

Dataset	Evaluation	Pre-trained	Data-based	Validity-based	Classifier-based
MNIST	$\texttt{Inv}(\downarrow)(\times 10^{-5})$	1.1×10^4	8.0 ± 2.2	6.4 ± 0.8	$\mathbf{5.2 \pm 3.7}$
(8 epochs)	$\texttt{IS}(\uparrow)$	7.82	$\mathbf{7.20 \pm 0.08}$	7.19 ± 0.04	7.16 ± 0.04
CIFAR-10	$\texttt{Inv}(\downarrow)(\times 10^{-3})$	1.3×10^2	$\mathbf{7.5 \pm 1.1}$	7.6 ± 1.0	11.6 ± 1.0
(30 epochs)	$\texttt{FID}(\downarrow)$	36.2	34.8 ± 1.5	34.8 ± 1.4	$\mathbf{33.2 \pm 0.6}$
STL-10	$\texttt{Inv}(\downarrow)(\times 10^{-4})$	6.2×10^2	8.8 ± 4.5	$\mathbf{7.7 \pm 1.3}$	11.6 ± 3.6
(40 epochs)	$\texttt{FID}(\downarrow)$	79.1	77.8 ± 2.2	$\mathbf{77.0 \pm 2.3}$	77.2 ± 1.5



(a) MNIST (b) CIFAR-10 (c) STL-10

Figure 1: Invalidity during data forgetting when forgetting label zero. Mean and standard errors are plotted for five random seeds. Standard errors may be too small to spot. Invalidity drops quickly at the beginning of data forgetting, and different algorithms are highly comparable to each other.

4 Experiments

In this section, we aim to answer the following questions.

- How well can the algorithms in Section 3 forget samples in practice?
- Can these algorithms be used to de-bias pre-trained models?
- Can these algorithms be used to understand training data?

In this section, we examine these questions by focusing on several real-world image datasets, including MNIST (28×28) [LeCun et al., 2010], CIFAR (32×32) [Krizhevsky et al., 2009], CelebA (64×64) [Liu et al., 2015] and STL-10 (96×96) [Coates et al., 2011] datasets. In Section 4.1, we investigate how well these algorithms can forget samples with a specific label. In Section 4.2, we investigate how well these algorithms can de-bias pre-trained models and improve generation quality. In Section 4.3, we use these algorithms to understand training data through the lens of data forgetting.

The pre-trained model for each dataset is a DCGAN [Radford et al., 2015] trained for 200 epochs (see details in Appendix D). We use one NVIDIA 3080 GPU to train these models and run experiments.

Evaluation Metrics: invalidity and generation quality. The invalidity is defined as the mass of the generation distribution on the forgetting set Ω: $\texttt{Inv}(p_G) = \int_{x \in \Omega} p_G(x) dx$. In practice, we measure invalidity by generating 50K samples and computing the fraction of these samples that fall into Ω.

The generation quality is measured in Inception Score (`IS`) [Salimans et al., 2016] and Frechet Inception Distance (`FID`) [Heusel et al., 2017]. Higher `IS` or lower `FID` indicates better quality. We compute `IS` for grey-scale images and `FID` for RGB images. When measuring quality, we compute `IS` or `FID` between 50K generated samples and $X \cap \bar{\Omega}$. Therefore, this score is not comparable with the score w.r.t. the pre-trained model if the forgetting set includes samples in the training set, such as samples with a specific label in Section 4.1. Detailed setup is in Appendix D.

4.1 Forgetting Labels

Question. How well can the algorithms in Section 3 forget samples in practice?

Methodology. We investigate how well the proposed algorithms can forget samples with a specific label y. In the data-based setting (Alg. 1), we express this as $\Omega = \{x \in X : \texttt{label}(x) = y\}$. In the validity-based setting (Alg. 2), we express this by setting $\mathbf{v}(x) = 1\{\arg\max_i \texttt{logit}(x)_i \neq y\}$,

Table 2: Study on the effect of T in Alg. 2 when the total number of queries is fixed. R refers to the number of epochs of data forgetting. A large T may lead to worse invalidity.

T	MNIST			CIFAR-10			STL-10		
	R	Inv(↓)	IS(↑)	R	Inv(↓)	FID(↓)	R	Inv(↓)	FID(↓)
400	20	$\mathbf{0.0 \times 10^{-4}}$	7.10	75	$\mathbf{0.45 \times 10^{-2}}$	35.1	100	1.0×10^{-3}	**75.1**
1000	8	0.6×10^{-4}	**7.19**	30	0.76×10^{-2}	34.8	40	$\mathbf{0.8 \times 10^{-3}}$	77.0
2000	4	2.8×10^{-4}	7.11	15	1.00×10^{-2}	**31.9**	20	1.0×10^{-3}	**75.1**

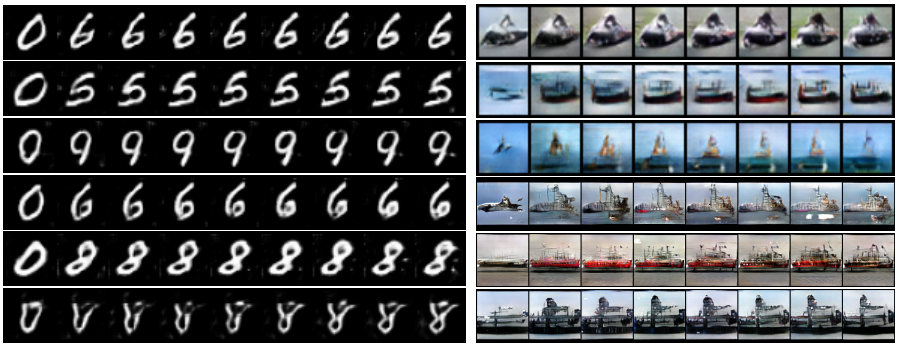


Figure 2: Visualization of the data forgetting process of invalid samples when forgetting label zero. The first column is generated by the pre-trained generator, and the i-th column is generated after $k \cdot (i-1)$ epochs of data forgetting. Left: MNIST with $k = 1$. Right: top is CIFAR-10 and bottom is STL-10, both with $k = 4$ and label zero being `airplanes`. We can see samples associated with invalid labels are gradually pushed to other labels, but a high-level visual similarity is kept.

where `logit` is the output of the softmax layer of a pre-trained label classifier [Chen, 2020]. In the classifier-based setting (Alg. 3), we set $\mathbf{f}(x) = 1 - \texttt{logit}(x)_y$.

In Table 1, we compare invalidity and generation quality among different algorithms and datasets when we forget label 0. We plot invalidity during data forgetting in Fig. 1. We also compare invalidity after one epoch of data forgetting in Appendix E.1.1. Mean and standard errors for 5 random runs are reported. Results for different hyper-parameters and forgetting other labels are in Appendix E.

Results. We find all the algorithms in Section 3 work quite well with a much fewer number of epochs used for training the pre-trained model (which is 200). These algorithms are generally comparable. Therefore, we conclude that the simplest data-based algorithm is good enough to forget samples when those training samples to be forgotten ($X \cap \Omega$) can characterize the forgetting set (Ω) well.

We also find invalidity rapidly drops after only one epoch of data forgetting, indicating these algorithms are very efficient in penalizing invalidity. While different algorithms perform better on different datasets, they are highly comparable with each other. The reason why the classifier-based algorithm performs the best on MNIST is possibly that the label classifier on MNIST is almost perfect so its gradient information is accurate.

Visualization. We sample latents $z \sim \mathcal{N}(0, I)$ and choose those corresponding to invalid samples, i.e. $G_0(z) \in \Omega$ where G_0 is the pre-trained generator. We select visually good $G_0(z)$ for demonstration. We visualize $G(z)$ during data forgetting in Fig. 2, and more visualizations are in Appendix E.3. This demonstrates how the latent space is manipulated: the label to be forgotten is gradually pushed to other labels, and there is high-level visual similarity between the final $G(z)$ and the original $G_0(z)$.

Effects of other hyper-parameters. In Table 2, we compare different T (#queries after each epoch) in the validity-based forgetting algorithm (Alg. 2). We fix the total number of queries by setting $T \times$#epochs to be a constant. Results indicate that a large T may lead to worse invalidity, and there is trade-off between invalidity and quality when setting T to be small or moderate.

In Appendix E.1.3, we compare different λ (hyperparameter in (2)) in the classifier-based forgetting algorithm (Alg. 3). We find there exists a clear trade-off between invalidity and quality when alternating λ: a larger λ tends to produce better quality, and a smaller λ tends to have better invalidity.

Forgetting multiple sets. We then investigate how well the proposed algorithms can generalize to multiple forgetting sets with methods in Section 3.4. We focus on the CelebA dataset [Liu et al.,

Table 3: Invalidity and generation quality of different forgetting algorithms on forgetting a combination of attributes within CelebA. There is a significant drop of invalidity, indicating that different forgetting algorithms can all generalize to multiple forgetting sets.

Evaluation	Pre-trained	Epochs	Data-based	Validity-based	Classifier-based
Inv(↓)	1.66×10^{-3}	1	9.0×10^{-4}	7.6×10^{-4}	$\mathbf{7.0 \times 10^{-4}}$
Inv(↓)	1.66×10^{-3}	5	$\mathbf{3.8 \times 10^{-4}}$	6.8×10^{-4}	6.8×10^{-4}
FID(↓)	36.4	5	29.3	29.9	**27.9**

Table 4: Invalidity after de-biasing boundary artifacts of generated MNIST samples. We run the validity-based forgetting algorithm (Alg. 2) for 4 epochs. The invalidity drops significantly, and a small or moderate T leads to slightly lower (better) invalidity.

	Pre-trained	$T = 5$K	$T = 10$K	$T = 20$K	$T = 40$K	$T = 80$K
Margin = 1	3.1×10^{-3}	$\mathbf{6.0 \times 10^{-5}}$	8.0×10^{-5}	2.0×10^{-4}	2.0×10^{-4}	7.0×10^{-4}
Margin = 2	1.1×10^{-3}	1.6×10^{-4}	$\mathbf{4.0 \times 10^{-5}}$	6.0×10^{-5}	3.2×10^{-4}	2.8×10^{-4}

Table 5: Invalidity and generation quality after de-biasing label biases of generated samples from different datasets. We use the classifier-based forgetting algorithm (Alg. 3). The arrow means improvement from the pre-trained model to after data forgetting. There is a clear improvement of generation quality, indicating the proposed algorithm can help GANs generate better samples.

τ	MNIST (8 epochs, $\lambda = 0.8$)		CIFAR-10 (30 epochs, $\lambda = 0.9$)	
	Inv(↓)	IS(↑)	Inv(↓)	FID(↓)
0.3	$8.19 \times 10^{-4} \to 2.60 \times 10^{-4}$	$7.82 \to \mathbf{8.10}$	$5.79 \times 10^{-4} \to 2.20 \times 10^{-4}$	$36.2 \to 27.1$
0.5	$2.07 \times 10^{-2} \to 1.70 \times 10^{-2}$	$7.82 \to 7.92$	$2.28 \times 10^{-2} \to 1.67 \times 10^{-2}$	$36.2 \to \mathbf{26.6}$
0.7	$1.35 \times 10^{-1} \to 1.22 \times 10^{-1}$	$7.82 \to 7.95$	$1.72 \times 10^{-1} \to 1.49 \times 10^{-1}$	$36.2 \to 26.8$

2015], which has 40 labeled attributes. We use proposed algorithms to forget a combination of these attributes: $\Omega_1 = \{\texttt{Black_hair and Blurry}\}$, $\Omega_2 = \{\texttt{Brown_hair and Wear_eyeglasses}\}$, and $\Omega = \Omega_1 \cup \Omega_2$. These attributes are randomly selected from those easy to capture. See detailed setup in Appendix E.4. Results after 1 or 5 epochs are reported in Table 3. Consistent with results on forgetting just one label, all algorithms can reduce invalidity and retain generation quality and are comparable, while the classifier-based algorithm achieves the best invalidity after one epoch.

4.2 Model De-biasing

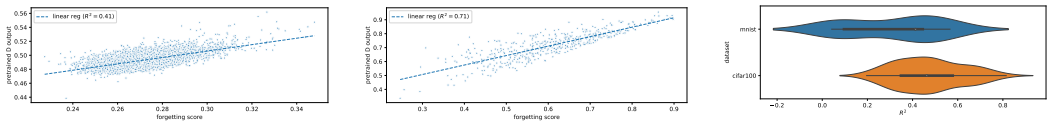
There can be different artifacts in GAN generated samples, and these could harm the overall generation quality. These artifacts may not exist in training samples, but are caused by inductive biases of the model, and become obvious after training. We can post-edit a pre-trained model to remove these artifacts, which we call *model de-biasing*. In this section, we investigate how well Alg. 2 and Alg. 3 apply to this task. We assume training samples are not biased so Alg. 1 does not apply to de-biasing.

To use these algorithms for de-biasing, we assume the target artifact or bias can be automatically detected by a classifier $\mathbf{f}$ or a validity function $\mathbf{v}$. Specifically, we survey two kinds of biases: boundary artifacts and label biases.

Boundary artifacts. A GAN trained on MNIST might generate samples that have numerous white pixels on the boundary (see Appendix F.1). We call this phenomenon the *boundary artifact*. We use the validity-based algorithm (Alg. 2) to de-bias boundary artifacts. The validity function is defined as $\mathbf{v}(x) = 1\{\sum_{(i,j)\in\text{boundary pixels}} x_{ij} < \tau_b\}$, where boundary pixels are those within a certain margin to the boundary, and threshold τ_b satisfies no training image is invalid.

Results are reported in Table 4. It is clear that the invalidity reduces in order after data forgetting, indicating boundary artifacts are largely removed. Consistent with Table 2, a small or moderate T leads to better results. We visualize samples before and after de-biasing in Appendix F.1.

Label biases. Neural networks may generate visually smooth but labelally ambiguous samples [Kirichenko et al., 2020], e.g. samples that look like multiple objects (see Appendix F.2). We call this phenomenon the *label bias*. We use the classifier-based algorithm (Alg. 3) to de-bias label biases. The classifier is defined as $\mathbf{f}(x) = 1 - \text{Entropy}(\texttt{logit}(x))/\log(\texttt{\#classes})$, where the `logit`



(a) MNIST (0) (b) CIFAR-100 (`aquarium_fish`) (c) Distribution of R^2

Figure 3: (a) and (b) Forgetting scores of invalid training samples ($\mathcal{FS}(x)$) versus the pre-trained discriminator outputs of them ($D_0(x)$). There is positive correlation between these two scores, indicating on-manifold samples are easier to forget. (a) Forgetting 0 in MNIST. (b) Forgetting `aquarium_fish` in CIFAR-100. (c) Distributions of R^2 scores of linear regression between $\mathcal{FS}(x)$ and $D_0(x)$ for all labels. The correlation in (a) and (b) is universal and stronger in CIFAR-100.

function is the same as in Section 4.1. Results are reported in Table 5. After de-biasing, we can improve the generation quality by a significant gap (~ 0.3 in `IS` and ~ 10 in `FID`). There is also a clear drop in terms of invalidity.

4.3 Understanding Training Data through the Lens of Data Forgetting

Large datasets can be hard to analyze. In this section, we investigate how data forgetting can help us understand these data. Specifically, we ask: which samples are easy or hard to forget?

In order to quantify the difficulty to forget a sample, we define the forgetting score $\mathcal{FS}$ to be the difference of discriminator outputs before and after data forgetting. Formally, let $x \in \Omega$ be a sample to forget, $\mathcal{M} = (G_0, D_0)$ be the pre-trained model, and $\mathcal{M}' = (G', D')$ be a model after data forgetting. Then, the forgetting score is $\mathcal{FS}(x) = D_0(x) - D'(x)$. A larger $\mathcal{FS}$ means it is easier to forget x.

To investigate **sample-level** forgetting difficulty, we forget a particular label at one time using Alg. 1. We then demonstrate scatter plots of forgetting scores $\mathcal{FS}(x)$ versus pre-trained discriminator outputs $D_0(x)$ for all samples x with this label. We also fit linear regression and report R^2 values (larger means stronger linear relationship). Scatter plots for some labels in MNIST and CIFAR-100 and distribution of R^2 for all labels are shown in Fig. 3. We also visualize the most and least difficult-to-forget samples in Appendix G. We find there is positive correlation between $\mathcal{FS}(x)$ and $D_0(x)$, indicating on-manifold (large $D_0(x)$) samples are easier to be forgotten, while off-manifold (small $D_0(x)$) ones are harder to be forgotten. This analysis further provides a way to investigate **label-level** forgetting difficulty. By averaging forgetting scores for samples associated with each label, we can survey which labels are easy or hard to forget in general. The results are in Appendix G. We find some labels are harder to forget than others.

4.4 Discussion: Relationship to Adversarial Samples

An adversarial sample for a classifier $\mathbf{f}$ and a sample x is another sample $\tilde{x} \approx x$ but $\mathbf{f}(\tilde{x}) \neq \mathbf{f}(x)$. Generating and defending these samples have become one of the most important directions of deep learning [Goodfellow et al., 2014b, Madry et al., 2018]. In this section, we show a variant of Alg. 3 can potentially be used to define a specific type of adversarial samples. In detail, we fix the discriminator D and only update the generator G while running Alg. 3. Then, G is trained to fool D and the classifier $\mathbf{f}$ at the same time. Notice that fooling D means generating on-manifold (visually similar to training data) samples, and fooling $\mathbf{f}$ means finding adversarial samples of $\mathbf{f}$. By combining these objectives we can force G to produce on-manifold adversarial samples, which may be significant in many real-world applications. We visualize some samples in Appendix H.

5 Conclusion

In this paper, we propose a systematic framework for forgetting data from pre-trained generative models. We provide three different algorithms for GANs that differ on how the samples to be forgotten are described. We provide theoretical results that data forgetting can be achieved. We then empirically investigate data forgetting on real-world image datasets, and show that our algorithms are capable of forgetting data while retaining high generation quality at a fraction of the cost of full re-training. One limitation or our paper is that the proposed framework only applies to unconditional generative models. It is an important future direction to define data forgetting and propose algorithms for conditional generative models, which are more widely used in downstream deep learning applications.

References

A. Abid, M. Farooqi, and J. Zou. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306, 2021.

A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*, 2017.

S. Asokan and C. Seelamantula. Teaching a gan what not to learn. *Advances in Neural Information Processing Systems*, 33:3964–3975, 2020.

D. Bau, S. Liu, T. Wang, J.-Y. Zhu, and A. Torralba. Rewriting a deep generative model. In *European Conference on Computer Vision*, pages 351–369. Springer, 2020.

Y. Cao and J. Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

A. Chen. Pytorch-playground. https://github.com/aaron-xichen/pytorch-playground, 2020.

A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

C. de Masson d'Autume, S. Mohamed, M. Rosca, and J. Rae. Training language gans from scratch. *Advances in Neural Information Processing Systems*, 32, 2019.

C. Donahue, J. McAuley, and M. Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014a.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

S. Hanneke, A. T. Kalai, G. Kamath, and C. Tzamos. Actively avoiding nonsense in generative models. In *Conference On Learning Theory*, pages 209–227. PMLR, 2018.

E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850, 2020.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

W. R. Inc. Mathematica, Version 13.0.0. URL https://www.wolfram.com/mathematica. Champaign, IL, 2021.

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

T. Kaneko and T. Harada. Blur, noise, and compression robust generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13579–13589, 2021.

C. Karakas, A. Dirik, E. Yalcinkaya, and P. Yanardag. Fairstyle: Debiasing stylegan2 with style channel manipulations. *arXiv preprint arXiv:2202.06240*, 2022.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

P. Kirichenko, P. Izmailov, and A. G. Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33:20578–20589, 2020.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033, 2020.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Q. Li, L. Mai, M. A. Alcorn, and A. Nguyen. A cost-effective method for improving and re-purposing large, pre-trained gans by fine-tuning their class-embeddings. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

S. Mo, M. Cho, and J. Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020.

X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

E. Perez, S. Huang, F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

G. K. Pitsilis, H. Ramampiaro, and H. Langseth. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*, 2018.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

K. Schwarz, Y. Liao, and A. Geiger. On the frequency bias of generative models. *Advances in Neural Information Processing Systems*, 34, 2021.

A. Sinha, K. Ayush, J. Song, B. Uzkent, H. Jin, and S. Ermon. Negative data augmentation. In *International Conference on Learning Representations*, 2020.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

A. Taha, A. Shrivastava, and L. S. Davis. Knowledge evolution in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12843–12852, 2021.

S. Tan, Y. Shen, and B. Zhou. Improving the fairness of deep generative models without retraining. *arXiv preprint arXiv:2012.04842*, 2020.

H. Thanh-Tung and T. Tran. Catastrophic forgetting and mode collapse in gans. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2020.

N. Thiem, M. Orescanin, and J. B. Michael. Reducing artifacts in gan audio synthesis. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1268–1275. IEEE, 2020.

S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8695–8704, 2020.

D. Warde-Farley and I. Goodfellow. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, 311:5, 2016.

J. Zhang, X. Chen, Z. Cai, L. Pan, H. Zhao, S. Yi, C. K. Yeo, B. Dai, and C. C. Loy. Unsupervised 3d shape completion through gan inversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1768–1777, 2021.

X. Zhang, S. Karaman, and S.-F. Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.

M. Zhao, Y. Cong, and L. Carin. On leveraging pretrained gans for generation with limited data. In *International Conference on Machine Learning*, pages 11340–11351. PMLR, 2020.

H. Zhou, A. Vani, H. Larochelle, and A. Courville. Fortuitous forgetting in connectionist networks. In *International Conference on Learning Representations*, 2021.

J. Zhu, Y. Shen, D. Zhao, and B. Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss limitations in the conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] We include proofs in the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide code in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify training details in the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We run with 5 random seeds and report error bars for the main experiment on forgetting labels.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We use one 3080 GPU.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide code in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We only run experiments on public datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]