Redactor: Model-Agnostic Targeted Disinformation Generation using Probabilistic Decision Boundaries

Geon Heo
KAIST
geon.heo@kaist.ac.kr

Steven Euijong Whang
KAIST
swhang@kaist.ac.kr

ABSTRACT

Information leakage is becoming a critical problem as various information becomes publicly available by mistake, and machine learning models train on that data to provide services. As a result, one's private information could easily be memorized by such trained models. Unfortunately, deleting information is out of the question as the data is already exposed to the Web or third-party platforms. Moreover, we cannot necessarily control the labeling process and the model trainings by other parties either. In this setting, we study the problem of *targeted disinformation* where the goal is to lower the accuracy of inference attacks on a specific target (e.g., a person's profile) only using *data insertion*. While our problem is related to data privacy and defenses against inference attacks, our techniques are inspired by targeted data poisoning attacks with some key differences. We show that our problem is best solved by finding the closest points to the target in the input space that will be labeled as a different class. Since we do not control the labeling process, we instead conservatively estimate the labels probabilistically by combining decision boundaries of multiple classifiers using data programming techniques. We also propose techniques for making the disinformation realistic. Our experiments show that a probabilistic decision boundary can be a good proxy for labelers, and that our approach outperforms other targeted poisoning methods when using end-to-end training on real-world datasets. Our approach also helps reduce information leakage caused by inference attacks and can scale to large data.

ACM Reference Format:
Geon Heo and Steven Euijong Whang. 2018. Redactor: Model-Agnostic Targeted Disinformation Generation using Probabilistic Decision Boundaries. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Information leakage is becoming a serious problem as personal data is being used to train machine learning (ML) models. As a striking example, Lee Luda [4] is an AI chat bot service that was shut down soon after its release because of its hate speech towards minorities and exposure of personal data including bank accounts and addresses. Another example is the Web [2] where life insurers can predict the life spans of their customers based on personal information leaked on the Web through various online activities. Recently, Flickr [1] has been in the news where photos uploaded in its early days are now used as training data for various ML computer vision models without author consent. Furthermore, there are various privacy threats on ML models including inference attacks [15, 30, 50] and reconstruction attacks [21, 22].

Unfortunately, one cannot simply delete the information if it is published on the Web or uploaded on a third-party platform, but can only dilute it. Even if the original data is deleted by request, there is no way to prevent someone from extracting that information elsewhere by attacking the model of the unknown third-party platform. Moreover, there is also no control over the model training process where anyone can train a model on the publicized data. Hence, conventional privacy techniques or inference attack defenses that require ownership of the data or model are out of the question. The only solution is to add more data to dilute one's information by reducing the model's accuracy and confidence of predictions on a certain target (e.g., personal profile) as shown in Figure 1. We assume this data will eventually be picked up automatically by

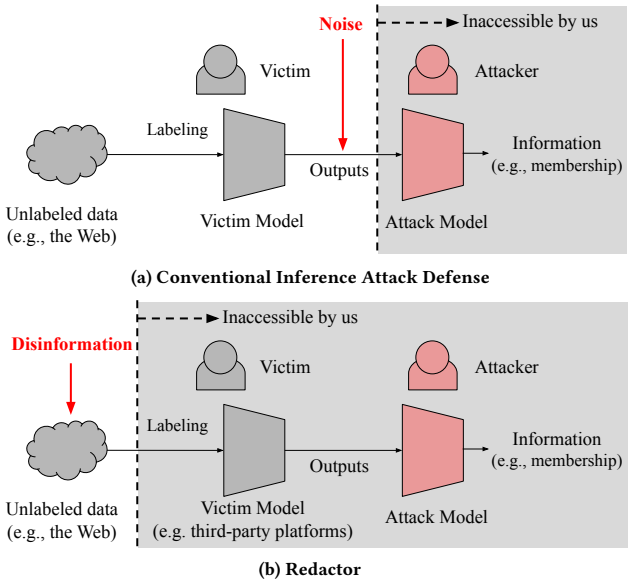


Figure 1: (a) Existing inference attack defenses (e.g., add noise to model's output) are not feasible when the victim models are owned by third parties. (b) In contrast, we assume the realistic setting where we can only add disinformation to the unlabeled training data, which is presumably labeled and used for model training by unknown model owners.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

crawlers for model training, which is a common assumption in the data poisoning literature explained below. An analogy is blacking out or redacting text where the reader knows there is some information, but cannot read it. We refer to this problem as *targeted disinformation* for data privacy purposes.

The most relevant work to disinformation generation is targeted poisoning [34, 49, 53, 60]. Recent techniques have been proposed mainly for attacking image classification models. Given a target image t and a base image b of a different class, the goal is to generate an image p that looks like b according to a labeler, but is classified the same as t by the model. Most techniques rely on a transfer learning [38] scenario where there is a pre-trained model that can be used to generate features in a fixed feature space. Here p can be made to be close to b in the input space (i.e., the pixels are similar), but close to t in the feature space. The labeler would label p to be the same as b, but this will confuse the model training, which thinks that p is the same as t. We would like to utilize such techniques to change the output of the unknown models (e.g., third-party platforms) and protect target instances from indirect privacy attacks. While transfer learning benefits certain applications (e.g., NLP or vision tasks), it is not always applicable, especially for structured data where there is no efficient and generally-accepted practice [9]. However, structured data is important because most personal information is stored in this format. In order to support structured data, we need to assume *end-to-end training* where the feature space is no longer fixed. Although existing techniques have also been extended for end-to-end training, we show their performances are not sufficient (see Sections 2 and 4.3).

While our approach is inspired by targeted poisoning attacks, the key difference is that we only utilize the input space to find the best disinformation that is close to the target, but labeled differently. How do we know the true label of the disinformation? Since we do not have access to the labelers, our key idea is to conservatively estimate human behavior using *probabilistic decision boundaries* produced by combining multiple possible classifiers. We adapt data programming [44–46], which combines multiple labeling functions into a label model that produces probabilistic labels. In our setting, we make the generative model produce the probability of an example having a class that is different than t's class. By limiting this probability to be above a tolerance threshold, we now have a conservative decision boundary. To illustrate our approach, Table 1 shows people records with attributes Education and Age. Let us say t is the target, and b_1 and b_2 are other examples. There are three binary surrogate models that produce the predictions $\hat{Y}_1$, $\hat{Y}_2$, and $\hat{Y}_3$. Let us say the probabilistic decision boundary labels an example as 0 (i.e., not in t's class) only if at least 60% (intentionally low for illustration purposes) of the surrogate models say so. As a result, we can only safely say that b_2 is not the same person as t. Given this decision boundary, we can generate a disinformation d_2 that is labeled the same as b_2, but has the closest distance to t as shown in Table 1 and Figure 2. We generate d_2 between t and b_2 using watermarking [13, 49] where a watermark of t is added to b_2 to generate d_2 using linear interpolation.

Our proposed system Redactor generates disinformation using probabilistic decision boundaries and base examples as shown in Table 1 and Figure 2. In addition, Redactor uses generative adversarial networks [27] to produce more disinformation examples like d_3 and d_4. Here d_4 happens to be even closer to t than d_2. Finally, Redactor ensures the disinformation is realistic by avoiding patterns of data that do not occur in the original data.

ID	Edu.	Age	$\hat{Y}_1$	$\hat{Y}_2$	$\hat{Y}_3$	$P(Y=0)$	Not the same as t?
Original Examples							
t	5	40	1	1	1	0	No
b_1	5	35	1	0	1	0.33	No
b_2	3	30	0	0	0	1	Yes
Generated Examples							
d_2	4	35	0	0	0	1	Yes
d_3	7	40	0	0	0	1	Yes
d_4	6	40	0	0	1	0.67	Yes

Table 1: A table of personal records and three predictions ($\hat{Y}_1$, $\hat{Y}_2$, and $\hat{Y}_3$) of binary surrogate models. The last two columns show how a conservative probabilistic decision boundary determines which examples are different than t by comparing the probability of the true label Y being zero, i.e., $P(Y=0)$, with a tolerance threshold of 0.6.

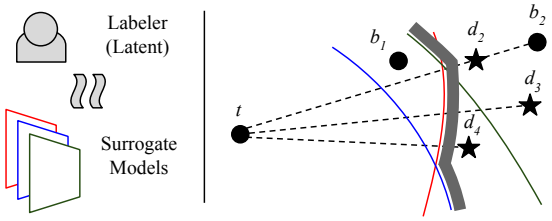


Figure 2: A labeler is approximated as a probabilistic model that combines surrogate models. The decision boundaries of surrogate models of Table 1 are shown as thin lines while the probabilistic decision boundary is the thick gray line. Using b_2, Redactor can generate the disinformation example d_2 that is close to t, but still labeled differently. In addition, Redactor can generate the fake realistic points d_3 and d_4 using a generative model where d_4 happens to be even closer to t.

We evaluate Redactor on real datasets using end-to-end model training. We show how the probabilistic decision boundaries are valid proxies of labelers and how Redactor outperforms other targeted poisoning attack baselines in terms of reducing the model's accuracy and confidence on the target and thus making the model's output more likely to change. We also show that Redactor reduces information leakage caused by inference attacks and can scale to large data. We release our code as a community resource [3].

In the rest of the paper, we present background (Section 2), an overview of Redactor and its main components (Section 3), experimental results (Section 4), and related work (Section 5).

2 BACKGROUND

We clarify the difference between a transfer learning scenario and end-to-end training scenario. We then explain why existing targeted poisoning attacks, which rely on transfer learning, fail to perform well in an end-to-end setting.

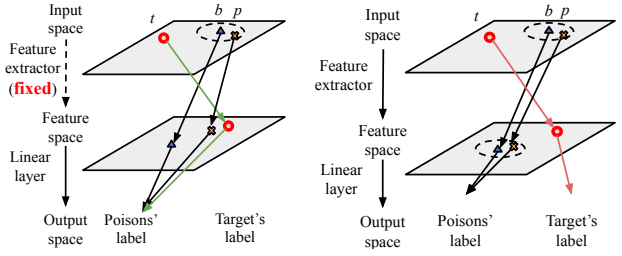
Comparison with Targeted Poisoning. Targeted poisoning attacks have the goal of flipping the predictions on specific targets to a certain class. A naïve approach is to add examples that are identical to the target, but with different labels. Unfortunately, such an approach would not work if one does not have complete control over the labeling process, which is unrealistic. Instead, the poison p needs to be different enough from the target to be labeled differently by any human. Yet, we also want p to be close to the target as well.

The state-of-the-art targeted poisoning attacks include Convex Polytope Attack (CPA) [60] and its predecessors [13, 49, 53], which also do not assume any control over the labeling and generate poison examples that are similar to the base examples, but have the same predictions as the target. Like our setting, these techniques are not involved in the model training itself, but generate poisoned examples that are presumably added to the training set. The goal is to generate examples close to the target in the feature space while being close to a base example in the input space as illustrated in Figure 3a. The common optimization solved by these techniques is:

$$p = \arg\min_{x} ||f(x) - f(t)||_2^2 + \beta||x - b||_2^2$$

where f maps inputs to features, t is the target, b is a base example that is in a different class than t, β balances the two objectives, and p is the generated poison example.

End-to-end Training. In end-to-end training, all layers of the model are trainable where any feature space that is not the input space may change after model training. Therefore, CPA's optimization may not be effective because any distance on the feature space corresponding to each layer can change arbitrarily. Figure 3b illustrates this point where the poison example p can still be close to the base example b on a feature space that is not the input space even after CPA's optimization. Although [60] suggests the extension of applying CPA on every layer of the network, it does not fundamentally solve the problem, especially on structured data. To demonstrate this point, we run the extended version of CPA on the AdultCensus dataset using a neural network (see more experimental results in Section 4.3) and generate a poison example p. We then observe how the relative L_2 distances between p and t change (Figure 4). As the model trains in a transfer learning scenario, the feature distance from p to t decreases on three different layers in the model (dotted lines). However, when the model trains end-to-end, the feature distance from p to t increases rapidly (solid lines), which means that the model no longer classifies p the same as t.



(a) Transfer Learning Scenario (b) End-to-end Training Scenario

Figure 3: In a transfer learning scenario (a), the feature space is fixed, making it possible to optimize on both the input and feature spaces. In an end-to-end scenario (b), however, the feature space may change after the model trains, so optimizing on the feature space may not be effective.

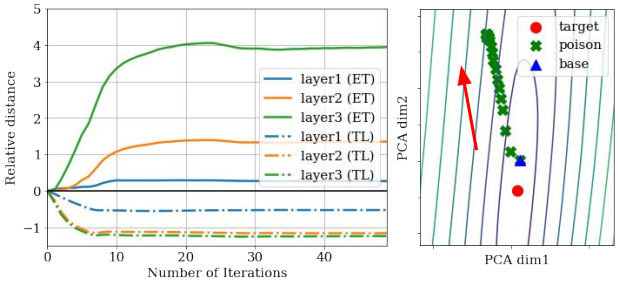


Figure 4: (Left) We run CPA [60] on the AdultCensus dataset and observe the relative L_2 distances from the initial points depicted in Figure 3. We measure distances from the poison example p to target t on possible feature spaces (different layers of a neural network). The transfer learning results (dotted lines) show how CPA is effective in reducing the distances from p to t, while the end-to-end training results (solid lines) show how it fails to do so. (Right) The target, poison, and base points on one of the feature spaces using PCA [55] for dimensionality reduction.

3 REDACTOR

3.1 Overview

We design an optimization problem of generating targeted disinformation for end-to-end training. We describe our objectives and introduce the overall process of Redactor. In end-to-end training, we can only utilize the input space and need to generate a disinformation that is as close as possible to the target example, but likely to be labeled as a different class from the target. Suppose that a human labeler has a mental decision boundary for labeling. In order to satisfy both conditions, the disinformation must be the closest point on the other side based on this decision boundary. Since we do not have control of the labeling and thus do not know the decision boundary, we propose to use surrogate models as a proxy for human labeling. This approach is inspired by ensemble techniques commonly used in forging black-box attacks [37, 60]. We do not assume that the surrogate models are highly accurate. However, when combining these models, we assume that we can find a *conservative decision boundary* that can confidently tell whether an example will be labeled differently than the target. An additional challenge is to make the disinformation as realistic as possible. For now, we assume there is a set $C_{real} \subseteq \mathbb{R}^D$ that conceptually contains all possible realistic candidates where D is the number of features. In Section 3.4, we propose techniques for generating realistic examples.

We now formulate our optimization problem as follows:

$$\begin{aligned} \min_{\{d_j\}} & \sum_{j=1}^{N_d} ||d_j - t||^2 \\ \text{s.t.} \quad \arg\max_{c} & \; M_c(\phi,\, d_j) \neq c_t \\ \max_{c \neq c_t} & \; M_c(\phi,\, d_j) \geq \alpha \\ d_j \in C_{real}, & \; \forall j \in [1 \ldots N_d] \end{aligned} \tag{1}$$

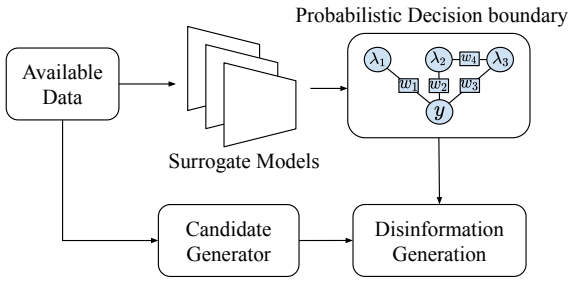


Figure 5: Redactor runs in three stages: surrogate model training, probabilistic decision boundary generation, and disinformation generation.

where $t \in \mathbb{R}^D$ is the target example, $d_j \in \mathbb{R}^D$ is the jth disinformation among a budget of N_d disinformations, $c \in [1...C]$ is a class that is not t's class c_t, $M_c(\phi,\ x)$ is the probabilistic generative model that combines surrogate models ϕ and returns the probability of an example x being in class c, and α is the tolerance threshold for the probabilistic decision boundary. We use common pre-processings where numeric features are normalized, and categorical features are converted to have numerical values using one-hot encoding.

Redactor generates disinformation in three stages: training surrogate models on the available data, generating a probabilistic decision boundary, and generating disinformation examples. This process is illustrated in Figure 5. In the next sections, we cover each component and its techniques in more detail.

3.2 Training Surrogate Models

When choosing surrogate models, it is useful to have a variety of models that can complement each other in terms of performance as they are not necessarily highly accurate. Similar strategies are used in data programming and ensemble learning [11, 23]. However, our goal is not necessarily improving the overall accuracy of the combined model, but ensuring a conservative probabilistic decision boundary. That is, there should be few false positives where a disinformation that is predicted to be on the other side of the target is actually labeled the same.

Another issue is that we may only have partial data for training surrogate models because the data is too large or unavailable. Indeed, if we are protecting personal information on the Web, it is infeasible to train a model on the entire Web data. However, we argue that we only need data that is in the vicinity of the target and contains some examples in different classes as well. We only require that the probabilistic decision boundary approximates the decision making around the target. For example, entity resolution [16, 20] also has this issue for large data, and there are solutions for resolving subsets of data of interest [8] in order to perform entity resolution in query time. We assume the same setting, except that we are training models instead. In Section 4.6, we show how Redactor can scale to large data by properly selecting partial data.

3.3 Probabilistic Decision Boundaries

We now explain how to generate a conservative probabilistic decision boundary for identifying examples that will very likely not be labeled the same as the target. We utilize multiple surrogate models and combine them into a single probabilistic model using data programming [44–46] techniques.

The data programming framework assumes that each labeling function (LF) can output one of the classes as a prediction or an abstained prediction (-1) if not confident enough. The abstained prediction is necessary for making the LFs as reliable as possible. We implement an LF using each surrogate model ϕ_i as follows:

$$\lambda(\phi_i,\ x) = \begin{cases} -1(Abstain) & \max_c \phi_i^{(c)}(x) \leq \beta/C \\ \arg\max_c \phi_i^{(c)}(x) & otherwise \end{cases}$$

where β is used to determine when to abstain, and ϕ_i outputs a C-dimensional probability vector.

We train a probabilistic generative model M with latent true labels Y using the label matrix $\Lambda_{\phi,x}$ where $\Lambda_{\phi,x}^{(i,j)} = \lambda(\phi_i,\ x_j)$:

$$P_w(\Lambda_\phi,\ Y) = Z_w^{-1}\ exp\left(\sum_{k=1}^{l} w^T\ Corr_k(\Lambda_\phi,\ y_k)\right)$$

$$\hat{w} = \arg\max_w\ \log \sum_Y P_w(\Lambda_\phi,\ Y)$$

$$M(\phi,\ d) = P_{\hat{w}}(Y|\ \Lambda_{\phi,d}).$$

Here $Corr$ indicates all possible correlations between LFs and the latent Y, Z_w^{-1} is the normalizing constant, and w has the weights of the generative model corresponding to each correlation.

We then use M as the probabilistic decision boundary. For each example d, M returns a probability distribution of classes. Then d is considered to be in a different class than the target t if the class with the maximum probability is not t's class, and the maximum probability is at least the tolerance threshold α.

3.4 Disinformation Generation

Given a target, we would like to find the closest possible points that would be labeled differently. Obviously we cannot use the target itself as it would not be labeled differently. Instead, we utilize the probabilistic decision boundary to find the closest point beyond the projected real decision boundary. We use watermarking [13, 31, 43, 49] techniques where a watermark of the target is added to the base example to generate disinformation using linear interpolations. While this approach works naturally for image data (i.e., the disinformation image is the same as the base image, but has a glimpse of the target image overlaid), structured data consists of numeric, discrete, and categorical features, so we need to perform watermarking differently. For numeric features, we can take linear interpolations. For discrete features that say require integer values, we use rounding to avoid outputting real numbers as a result of the interpolation. For categorical features, we choose the base's value or target's value, whichever is closer. More formally:

$$numeric:\ d^{(i)} = \gamma t^{(i)} + (1-\gamma)b^{(i)}$$

$$discrete:\ d^{(i)} = round(\gamma t^{(i)} + (1-\gamma)b^{(i)})$$

$$categorical:\ d^{(i)} = round(\gamma)t^{(i)} + round(1-\gamma)b^{(i)}$$

where d is the disinformation example, t is the target, b is a base example, $x^{(i)}$ is x's attributes corresponding to the feature index set i, $round(x) = \lfloor x + 0.5 \rfloor$, and $0 \leq \gamma \leq 1$.

In order to increase our chances of finding disinformation closer to the target, we can use GANs to generate more bases that are realistic and close to the decision boundary. Among possible GAN techniques for tabular data [7, 14, 40, 51, 57, 58], we extend the conditional tabular GAN (CTGAN) [57], which is the state-of-the-art method for generating realistic, but fake tabular data. CTGAN's key techniques are using mode-specific normalization to learn complicated column distributions and training-by-sampling to overcome imbalanced training data.

Making Examples Realistic. CTGAN does not guarantee that all constraints requiring domain knowledge are satisfied. For example, in the AdultCensus dataset, the marital status "Wife" means that the person is female, but we need to perform separate checking instead of relying on CTGAN. Our solution is to avoid certain patterns that are never seen in the original data. In our example, there are no examples where a Wife is a male, so we ignore all CTGAN-generated examples with this combination. This checking can be performed efficiently by identifying frequent feature pairs in the original data and rejecting any feature pair that does not appear in this list. In addition, we use clipping and quantization techniques to further make sure the feature values are valid.

3.5 Putting Everything Together

Algorithm 1 shows the overall algorithm of Redactor. We first select random base examples that are preferably close to the target, but obviously have different labels according to our judgement (Step 2). We then generate candidate disinformation examples using watermarking and a CTGAN (Steps 3–8). We also construct the probabilistic decision boundary by combining good-performing surrogate models into a probabilistic model (Steps 9–13). Finally, we return the disinformation examples that are on the other side of the decision boundary from the target (Steps 14–19).

Algorithm 1: Pseudo code for generating disinformation.

Input : Target example t, available data I, trained surrogate models ϕ, trained generator model G, number of disinformation examples N_d, number of generated samples N_{gen}, tolerance threshold α, abstain threshold β

Output : Disinformation examples R

// Generate candidate examples

1 $C_{real} \leftarrow [\,]$;

2 $B \leftarrow$ NearestExamples(I, t, N_d) $s.t.\ c_b \neq c_t$;

3 **for** i *in 0...r* **do**

4 $\quad \gamma \leftarrow i/r$;

5 $\quad C_{real}$.append(WaterMarking(B, t, γ));

6 $C_{GAN} \leftarrow$ G.generate($N_{gen}{}^{*}n$);

7 $C_{GAN} \leftarrow$ FilterUnrealisticRecord(C_{GAN}, I);

8 C_{real}.append(NearestExamples(C_{GAN}, t, N_{gen}));

// Decision boundary approximation

9 $\phi_{topK} \leftarrow$ SelectTopKmodels(I, ϕ, k);

10 **for** $\phi_i \in \phi$ **do**

11 $\quad \Phi_i \leftarrow \phi_i(I)$

12 $\Lambda \leftarrow$ LabelMatrixTransform(Φ, β);

13 $M \leftarrow$ TrainLabelModel(Λ, I);

// Generate disinformation examples

14 $R \leftarrow [\,]$;

15 **for** j *in 1...*N_d **do**

16 $\quad d_j \leftarrow \arg\min_{x \in C_{real}} ||x - t||^2\ s.t.\ M_c(x) \geq \alpha$;

17 $\quad R$.append(d_j);

18 $\quad C_{real}$.remove(d_j);

19 **return** R;

4 EXPERIMENTS

We evaluate Redactor and answer the following questions.

- Is a probabilistic decision boundary a good labeler proxy?
- How effective is Redactor's disinformation in reducing model accuracy and confidence?
- Can Redactor's disinformation reduce information leakage caused by inference attacks?
- How realistic is Redactor's disinformation to humans?
- Can Redactor scale to large data by using partial data?

4.1 Settings

Datasets. We use four real tabular datasets for binary and multi-class classification tasks. All the datasets contain people records whose information can be leaked. The last Diabetes dataset is large and thus used to demonstrate the scalability of our techniques.

- AdultCensus [35]: Contains 45,222 people examples and is used to determine if one has a salary of ≥$50K per year.
- COMPAS [6]: Contains 7,214 examples and is used to predict criminal recidivism rates.
- Epileptic Seizure Recognition (ESR) [5]: Contains 11,500 electroencephalographic (EEG) recording data and is used to classify five types of brain states including epileptic seizure.
- Diabetes [52]: A large dataset that contains 100,000 records of diabetes patients in 130 US hospitals between 1999–2008.

Target and Base Examples. For each dataset, we choose 10 targets per dataset randomly. For each target, we choose k nearest examples with different labels as the base examples to generate k watermarked disinformation examples.

Measures. To evaluate a probabilistic decision boundary, we use *precision*, which is defined as the portion of examples that are on the other side of the decision boundary from the target that actually have different ground truth labels. To evaluate a model's performance, we measure the *accuracy*, which is the portion of predictions that are correct, and use the *confidence* given by the model. For all measures, we always report percentages.

Models. We use three types of models: *surrogate models* for probabilistic decision boundaries, *victim models* to simulate inaccessible black-box models, and *attack models* that are used to perform inference attacks (only used in Section 4.4; also see Figure 1).

We use 18 surrogate models explained below and summarized in Table 2 . Although we could use more complex models, they would overfit on our datasets.

- Seven neural networks that have different combinations of the number of layers, the number of nodes per layer, and the activation function. We use the naming convention *s_nn_A_X-Y*, which indicates a neural network that uses the

Table 2: 18 surrogate model architectures and their individual Train, Test, and Cross Validation (CV) accuracies on the AdultCensus dataset. The CV accuracies are needed to select the top-k performing models.

	Surrogate Model	Train Acc.	Test Acc.	CV Acc.
s_nn	*tanh_5-2*	86.46	85.07	84.24
	relu_5-2	86.57	85.24	84.92
	relu_50-25	90.33	82.44	82.67
	relu_200-100	95.55	81.56	81.64
	relu_25-10	87.93	84.22	83.64
	log_5-2	85.63	85.26	84.50
	identity_5-2	84.84	84.74	84.79
s_tree	*dt_gini*	85.28	85.56	84.73
	dt_entropy	85.29	85.38	84.84
	rf_gini	85.08	85.21	84.93
	rf_entropy	85.16	85.37	84.96
s_svm	*rbf*	85.83	84.92	84.53
	linear	84.78	85.03	84.63
	polynomial	85.13	83.16	82.79
	sigmoid	81.24	82.11	82.22
others	*s_gb*	85.70	85.99	86.12
	s_ada	86.22	86.30	86.12
	s_logreg	84.90	84.86	84.76

activation function A (*tanh*, *relu*, *log*, and *identity*) and has X layers with Y nodes per layer.

- Two decision trees *s_tree* and two random forests (*s_rf*) using the Gini and Entropy purity measures.
- Four SVM models (*s_svm*) using the radial basis function (*rbf*), linear, polynomial, and sigmoid kernels.
- Three other models: gradient boosting (*s_gb*), AdaBoost (*s_ada*), and logistic regression (*s_logreg*).

For the victim models, we use a subset of Table 2 consisting of 13 models (four neural networks, four trees and forests, two SVMs, and three others), but with different numbers of layers and optimizers to clearly distinguish them from the surrogate models. For the attack models, we select nine of the smallest models having the fewest layers, depth, or number of tree estimators from Table 2. We choose small models because attack models train on a victim model's output and loss and need to be small to perform well. We use the same naming conventions as Table 2 except that the model names start with "a_" instead of "s_" as shown in Table 5 in Section 4.4.

Methods. We compare Redactor with three baselines: (1) *CPA* is the convex polytope attack extended to end-to-end training described in Section 2; (2) *GAN only* is Redactor using a CTGAN only; and (3) *WM only* is Redactor using watermarking only.

Other Settings. We set the abstain threshold β to 0.1. For all models, we set the learning rate to 1e-4 and the number of epochs to 1K. For CTGAN [57], we set the input random vector size to 100. We use PyTorch [41] and Scikit-learn [42], and all experiments are performed using Nvidia Titan RTX GPUs. We evaluate all models on separate test sets.

Table 3: Precision for probabilistic decision boundaries with different α tolerance thresholds (0.5–0.99) and taking a majority vote of the surrogate models (MV).

Group	0.5	0.7	0.9	0.95	0.99	MV
g_all	83.68	83.86	84.13	84.35	84.38	84.42
g_top-15	84.52	84.72	85.11	85.22	85.59	84.37
g_top-10	85.00	85.00	85.20	85.49	**86.28**	83.90
g_top-5	85.37	86.39	**87.95**	**88.74**	78.92	84.24
g_top-3	85.30	**87.18**	82.45	82.45	78.54	75.39
g_nn-only	81.74	81.79	82.08	82.32	82.66	84.43
g_tree-only	**85.44**	86.12	87.86	88.18	79.34	84.35
g_svm-only	82.96	82.96	82.96	82.96	64.20	**84.83**
g_others	85.13	86.94	80.33	80.33	77.27	75.39

4.2 Decision Boundary as a Labeler Proxy

We evaluate the probabilistic decision boundary precision in Table 3. We use cross validation accuracies to select the top-k performing surrogate models without knowledge of the test accuracies. We then use the following groups of surrogate models: *g_all* contains all the models, *g_top-k* contains the top-performing surrogate models, *g_nn_only* contains the neural network models, *g_tree_only* contains the tree models, *g_svm_only* contains the SVM models, and *g_others* contains the rest of the models. The table shows the probabilistic decision boundary's precision for different α tolerance thresholds. As α increases, the precision tends to increase except for model groups with fewer than five surrogate models. We observe that combining more surrogate models improves the precision as well, but only to a certain extent. Compared to taking a majority vote of surrogate models (MV), the precision of a probabilistic decision boundary is usually higher. In particular, using *top-5* combined with $\alpha = 0.95$ results in the best precision. We thus use this setting in the remaining sections.

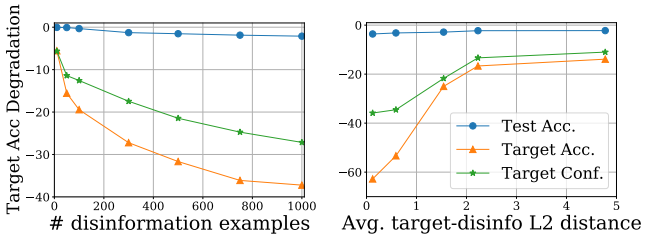
4.3 Disinformation Performance

We evaluate Redactor's disinformation in terms of how it reduces a victim model's accuracy and confidence on the AdultCensus, COMPAS, and ESR datasets in Table 4. For each dataset, we train the 13 victim models described in Section 4.1. We then generate disinformation for the targets (500 examples for AdultCensus, 50 for COMPAS, and 100 for ESR) and re-train the victim models on the dataset plus disinformation. As a result, Redactor reduces the performances more than the other baselines (especially CPA) without reducing the test accuracy significantly.

Using the same victim model setting, we also analyze how the number of disinformation examples and the distance between the target and disinformation impacts the disinformation performance. We first select 10 random target examples and vary the number of disinformation examples generated. Figure 6a shows how the average target accuracy and confidence of the 13 victim models decrease further as more disinformation examples are generated, but eventually plateaus. Next, we select 50 random target examples and generate disinformation. Then we cluster the targets by their average L_2 distances to their disinformation examples. We then plot each cluster in Figure 6b, which shows the average target accuracy and confidence degradation of the 13 models against the

Table 4: Average performance change of victim models on targets when generating disinformation examples on the AdultCensus, COMPAS, and ESR datasets. The number of inserted examples is about 1% of the entire dataset size.

		Overall Test Acc. Change	Target Acc. Change	Target Conf. Change
Adult Census	*CPA*	-0.27±0.52	-2.78±8.08	-1.97±8.60
	GAN only	-0.49±0.65	-16.67±13.72	-13.30±10.69
	WM only	-1.43±1.40	-28.89±12.78	-21.35±15.00
	Redactor	-1.99±1.73	**-37.22±13.20**	**-26.23±14.44**
COMPAS	*CPA*	-0.26±0.80	-0.56±5.39	-2.24±3.10
	GAN only	-0.14±0.72	-5.56±10.96	-2.30±3.31
	WM only	-2.31±2.08	-32.77±20.23	-21.68±13.26
	Redactor	-2.40±2.18	**-33.89±18.83**	**-23.93±14.37**
ESR	*CPA*	-0.43±4.27	-7.14±16.04	-2.59±5.75
	GAN only	-0.65±1.58	-8.57±15.74	-1.57±10.37
	WM only	-0.07±1.13	-34.29±12.72	-18.42±17.16
	Redactor	-0.11±0.89	**-35.71±13.97**	**-18.28±17.25**



(a) (b)

Figure 6: (a) As the number of disinformation examples increases, the target accuracy and confidence decrease significantly while the overall test accuracy decreases only by 2%. (b) As the distance to the target increases, we observe increasing trends as opposed to (a).

average target-disinformation L_2 distance of each cluster. As the disinformation is further away from a target, it becomes difficult to reduce the target's accuracy and confidence.

4.4 Defense Against Inference Attacks

Redactor can also defend against membership inference attacks (MIAs), which are the most popular inference attacks studied in the literature [50]. The goal of an MIA is to train an attack model that predicts if a specific example was used to train a victim model based on its predictions and loss values. As explained in Section 4.1, we use 9 independent models in Table 2 with different hyperparameters for attacking the trained victim models. We use the AdultCensus dataset and select 10 target examples. Table 5 shows the MIA performances with and without 200 disinformation examples using the 9 attack models. For each scenario, we specify the attack model's overall F_1 score and average target accuracy. We use the F_1 score just for this experiment to address the class imbalance of membership versus non-membership. Each experiment is repeated seven times. The less accurate the attack model, the better the privacy of the target. As a result, the overall F_1 score of the attack model does not change much, but the target accuracy decreases significantly

Table 5: Using Redactor's disinformation to defend against MIAs using attack models. For 10 target examples, a total of 200 disinformation examples are generated. For each model, we show how the disinformation changes its performances.

	Without Disinfo.		With Disinfo.		
Attack Model	Overall F1 score	Target Acc.	Overall F1 score	Target Acc.	Target Acc. Change
a_tanh_5-2	58.96	78.57	59.41	65.71	-12.86
a_relu_5-2	61.18	82.86	61.33	71.43	-11.43
a_identity_5-2	59.19	75.71	59.04	65.71	-10.00
a_dt_gini	52.02	73.33	51.04	46.67	-26.66
a_dt_entropy	52.22	60.00	51.66	43.33	-16.67
a_rf_gini	52.20	65.00	52.03	51.67	-13.33
a_rf_entropy	51.78	55.71	51.86	45.71	-10.00
a_ada	53.85	61.43	52.74	54.29	-7.14
a_logreg	61.30	80.00	61.20	70.00	-10.00
Average	55.86	70.29	55.59	57.17	-13.12

Table 6: Using Redactor's disinformation to defend against MIAs that do not use attack models. The other conditions are identical to Table 5. We use ROC AUC to measure the average performance over changing thresholds.

	Without Disinfo.		With Disinfo.		
Threshold MIA Type	Overall AUC	Target AUC	Overall AUC	Target AUC	Target AUC Change
Loss MIA [59]	49.93	88.75	49.85	63.96	-24.79
Conf. MIA [47]	50.55	88.75	50.52	69.79	-18.96

(by up to 26%) due to the disinformation. Furthermore some target accuracies drop to around 50%, which means the classification is almost random. In addition, Table 6 shows the bigger degradation in Loss MIA [59] and Conf. MIA [47], which do not require the attack models, but only threshold values on the loss and confidence score, respectively.

4.5 Realistic Examples

We perform a comparison of our disinformation with real data to see how realistic it is. Recall in Section 3.4 that we filter out examples that contain feature pair patterns that do not occur in the original data. Table 7 shows a representative disinformation example D (among many others) that was generated using our method along with the target T and the target's nearest examples NN. To see if the disinformation is realistic, we conduct a poll asking 11 human workers to correctly identify five disinformation and five real examples. As a result, the average accuracy is 53%, and the accuracies for identifying disinformation and real examples are 40% and 65%, respectively. We thus conclude that humans cannot easily distinguish our disinformation from real data, and that identifying disinformation is harder than identifying real examples.

4.6 Scalability

If the dataset is large or not fully available, Redactor can still run on partial data as explained in Section 3.2. We evaluate Redactor

Table 7: Comparison of disinformation D of target T with T's nearest neighbors NN using the AdultCensus dataset.

	Age	Workclass	Education	Marital status	Occupation	Relationship	Race	Gender	Capital gain	Hrs/week	Country	Income
T	38	Private	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	40	US	≤50K
D	43	Private	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	7676	40	US	>50K
NN	41	Private	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	40	US	≤50K
	37	Private	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	40	US	≤50K
	52	Private	HS-grad	Never-married	Machine-op-inspct	Not-in-family	White	Male	0	45	US	>50K
	36	Private	HS-grad	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	7298	40	US	>50K

Table 8: Evaluation of the two partial data strategies on the Diabetes dataset. For different data sizes (n), we show the average runtime for disinformation generation per target in seconds (Time), average local accuracy (Local Acc.), and average target accuracy change (Δ Acc.).

	Dist. Known			Dist. Unknown		
n	Time (s)	Local Acc.	Δ Acc.	Time (s)	Local Acc.	Δ Acc.
1k	59.72	58.82	-30.00	47.35	66.64	-31.67
3k	100.52	68.69	-30.00	79.85	69.30	-30.00
5k	125.76	70.40	-31.67	109.28	69.84	-27.77
7k	162.78	72.30	-26.67	122.68	70.42	-20.37
All	2,043.1	70.67	-36.67	2,043.1	70.67	-36.67

on the Diabetes dataset by selecting 10 random targets, training the 18 surrogate models on nearest neighbors of the targets using Euclidean distance, and generating 200 disinformation examples per target. We use two strategies for selecting the partial data: (1) *Dist. Known*: We assume the entire distribution is known and collect n nearest neighbors of targets following this distribution (i.e., we effectively take a uniform sample of the entire data that is closest to the target) and (2) *Dist. Unknown*: We assume the data distribution is unknown and collect nearest neighbors of the target until we have n examples. If there is a severe class imbalance, one can optionally collect more neighbors, but we do not run into this issue.

In Table 8, we compare the following for different n values: (1) the average runtime for disinformation generation per target, (2) the average local accuracy, which is the average accuracy of surrogate models on the 10K nearest neighbors of each target, and (3) the average target accuracy change. As a result, when n is at least 3,000 (3% of the entire data), the runtime improves by >20x, while the average local accuracy and target accuracy change are comparable to the results using the entire data (All). In addition, utilizing the data distribution sometimes gives worse results than not due to the adjustment of class ratios of nearest neighbors to follow the entire distribution. Hence, using partial data without knowing the entire data distribution can be sufficient for effective disinformation.

5 RELATED WORK

Redactor is related to multiple disciplines, and we explain why it solves a novel problem. The objective of generating disinformation is related to data privacy and deletion, although the problem setting is different where there is a single target to protect, and there is no control over the model or data. The techniques of Redactor are closely related to data poisoning attacks, although we assume the more common setting of end-to-end training instead of a transfer learning setting. Finally, Redactor generates realistic disinformation primarily for structured data.

Data Privacy, Data Deletion, and Disinformation. Data privacy is a broad discipline of protecting one's personal information within data. The most popular approach is differential privacy [17–19] where random records are added to a database to lower the chance of information leakage. In comparison, we solve a subproblem of data privacy in ML where there is no control over the training data, and the only way to improve one's privacy is to add disinformation.

A related problem is data deletion where the goal is to make a model forget about certain data. Data deletion has been recently studied when using k-means clustering [24], non-iterative ML models [48], L_2-regularized linear regression [29, 32], neural networks (scrubbing [25, 26] and forgetting [10, 28, 56]). Most of these techniques assume that the data or model can be changed at will. In comparison, we only assume that data can be added and that models may be trained with the new data at some point. We also do not assume label information, but do not control the end model either.

The concept of disinformation is not new and has been studied in different contexts. A work on data leakage detection [39] uses disinformation to determine whether any information has been leaked when data is distributed. A work on entity resolution [54] proposes optimization techniques for lowering the entity resolution accuracy while using a limited budget for generating disinformation. In comparison, Redactor focuses on obfuscating information in ML models for data privacy.

Data Poisoning. Targeted poisoning attacks [34, 49, 53, 60] have the goal of flipping the predictions of specific targets to certain classes. Clean-label attacks [49, 53, 60] have been proposed for neural networks to alter the model's behavior on a specific test instance by poisoning the training set without having any control over the labeling. Convex Polytope Attack (CPA) [60] covers various structures of neural networks, which is different from other techniques. The goal is to generate examples close to the target in the feature space while close to a base in the input space. Both of these techniques primarily target a transfer learning setting whereas Redactor is designed for end-to-end training.

Exploratory attacks are used to extract information from models, and various defenses have been proposed. The dominant attack most related to our work is the membership inference attack, and many defenses [33, 36, 47] have been proposed. However, most works assume access to the victim's model. For example, MemGuard [33] is a state-of-the-art defense that adds noise to the

model's output to drop the attack model's performance. Other techniques include adding a regularizer to the model's loss function [36] and applying dropout or model stacking techniques [47]. However, such model modifications are not possible in our setting where we assume no access to the model.

Tabular Data Generation. Generating fake tabular data is becoming a major area in GAN research [12, 40, 57, 58]. CTGAN [57] generates realistic data, but obviously does not necessarily satisfy all necessary constraints as they require domain knowledge. FakeTables [12] focuses on satisfying functional dependencies when normalizing tables. LowProFool [7] generates adversarial examples that are imperceptible by only modifying relatively unnoticeable features. Redactor can utilize any of these techniques for generating realistic base examples.

6 CONCLUSION

We proposed effective targeted disinformation methods for black-box models on structured data where there is no access to the labeling or model training. We explained why an end-to-end training setting is important and that existing poisoning attacks that rely on a transferable learning setting do not perform well. We then presented Redactor, which is designed for end-to-end training where it generates a conservative probabilistic decision boundary to emulate labeling and then generates realistic disinformation examples that reduce the target's accuracy and confidence the most. Our experiments showed that Redactor generates disinformation more effectively than other poisoning attacks, defends against membership inference attacks, generates realistic disinformation, and scales to large data.

REFERENCES

[1] [n.d.]. How Photos of Your Kids Are Powering Surveillance Technology. https://www.nytimes.com/interactive/2019/10/11/technology/flickr-facial-recognition.html. Accessed Apr. 10th, 2022.

[2] [n.d.]. Insurers Test Data Profiles to Identify Risky Clients. https://www.wsj.com/articles/SB10001424052748704648604575620750998072986. Accessed Apr. 10th, 2022.

[3] [n.d.]. Redactor Github repository. https://github.com/anonymous-5381/Redactor/. Accessed April 11th, 2022.

[4] [n.d.]. South Korean AI chatbot pulled from Facebook after hate speech towards minorities. https://www.theguardian.com/world/2021/jan/14/time-to-properly-socialise-hate-speech-ai-chatbot-pulled-from-facebook. Accessed Apr. 10th, 2022.

[5] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. 2001. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E* 64, 6 (2001), 061907.

[6] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine bias: There's software used across the country to predict future criminals. And its biased against blacks. ProPublica.

[7] Vincent Ballet, Xavier Renard, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki. 2019. Imperceptible Adversarial Attacks on Tabular Data. *CoRR* abs/1911.03274 (2019).

[8] Indrajit Bhattacharya, Lise Getoor, and Louis Licamele. 2006. Query-time entity resolution. In *SIGKDD*. 529–534.

[9] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2021. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889* (2021).

[10] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2019. Machine Unlearning. *CoRR* abs/1912.03817 (2019).

[11] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[12] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and V. S. Subrahmanian. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data. In *IJCAI*. 2074–2080.

[13] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR* abs/1712.05526 (2017).

[14] Edward Choi, Siddharth Biswal, Bradley A. Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. 2017. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. In *MLHC*, Vol. 68. PMLR, 286–305.

[15] Christopher A Choquette Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2020. Label-Only Membership Inference Attacks. *arXiv preprint arXiv:2007.14321* (2020).

[16] Peter Christen. 2012. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer.

[17] Cynthia Dwork. 2011. A firm foundation for private data analysis. *Commun. ACM* 54, 1 (2011), 86–95.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, Vol. 3876. Springer, 265–284.

[19] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[20] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 1–16.

[21] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *CCS*. 1322–1333.

[22] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 17–32.

[23] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *icml*, Vol. 96. Citeseer, 148–156.

[24] Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. 2019. Making AI Forget You: Data Deletion in Machine Learning. In *NeurIPS*. 3513–3526.

[25] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *CVPR*. 9301–9309.

[26] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2020. Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observations. In *ECCV*. 383–398.

[27] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.

[28] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. 2021. Amnesiac Machine Learning. In *AAAI*. 11516–11524.

[29] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. 2020. Certified Data Removal from Machine Learning Models. In *ICML*. 3832–3842.

[30] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019. LOGAN: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies* 2019, 1 (2019), 133–152.

[31] Dorjan Hitaj and Luigi V Mancini. 2018. Have you stolen my model? evasion attacks against deep neural network watermarking techniques. *arXiv preprint arXiv:1809.00615* (2018).

[32] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Y. Zou. 2020. Approximate Data Deletion from Machine Learning Models: Algorithms and Evaluations. *CoRR* abs/2002.10077 (2020).

[33] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. 2019. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *CCS*. ACM, 259–274.

[34] Mehran Mozaffari Kermani, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K. Jha. 2015. Systematic Poisoning Attacks on and Defenses for Machine Learning in Healthcare. *IEEE J. Biomed. Health Informatics* 19, 6 (2015), 1893–1905.

[35] Ron Kohavi. 1997. Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. *KDD* (09 1997).

[36] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. 2021. Membership Inference Attacks and Defenses in Classification Models. In *CODASPY*. ACM, 5–16.

[37] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into Transferable Adversarial Examples and Black-box Attacks. In *ICLR*.

[38] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.

[39] Panagiotis Papadimitriou and Hector Garcia-Molina. 2011. Data Leakage Detection. *IEEE Trans. Knowl. Data Eng.* 23, 1 (2011), 51–63.

[40] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data Synthesis based on Generative Adversarial Networks. *Proc. VLDB Endow.* 11, 10 (2018), 1071–1083.

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. In *NIPS Autodiff Workshop*.

[42] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of Machine Learning Research* 12 (2011), 2825–2830.

[43] Erwin Quiring, Daniel Arp, and Konrad Rieck. 2018. Forgotten siblings: Unifying attacks on machine learning and digital watermarking. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 488–502.

[44] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proc. VLDB Endow.* 11, 3 (2017), 269–282.

[45] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason A. Fries, Sen Wu, and Christopher Ré. 2020. Snorkel: rapid training data creation with weak supervision. *VLDB J.* 29, 2-3 (2020), 709–730.

[46] Alexander J. Ratner, Stephen H. Bach, Henry R. Ehrenberg, and Christopher Ré. 2017. Snorkel: Fast Training Set Generation for Information Extraction. In *SIGMOD*. 1683–1686.

[47] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *NDSS*.

[48] Sebastian Schelter. 2020. "Amnesia" - Machine Learning Models That Can Forget User Data Very Fast. In *CIDR*.

[49] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *NeurIPS*. 6106–6116.

[50] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[51] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. 2017. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *NeurIPS*. 3308–3318.

[52] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. 2014. Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records. *BioMed Research International* (2014).

[53] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daumé III, and Tudor Dumitras. 2018. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *27th USENIX Security Symposium*. 1299–1316.

[54] Steven Euijong Whang and Hector Garcia-Molina. 2013. Disinformation techniques for entity resolution. In *CIKM*. 715–720.

[55] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2, 1-3 (1987), 37–52.

[56] Yinjun Wu, Edgar Dobriban, and Susan B. Davidson. 2020. DeltaGrad: Rapid retraining of machine learning models. In *ICML*. 10355–10366.

[57] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. In *NeurIPS*. 7333–7343.

[58] Lei Xu and Kalyan Veeramachaneni. 2018. Synthesizing Tabular Data using Generative Adversarial Networks. *CoRR* abs/1811.11264 (2018).

[59] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 268–282.

[60] Chen Zhu, W. Ronny Huang, Hengduo Li, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2019. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In *ICML*. 7614–7623.