Influence Attributions Can Be Systematically Altered by Model Manipulation

Chhavi Yadav*
UC San Diego

Ruihan Wu*
UC San Diego

Kamalika Chaudhuri
UC San Diego

Abstract

Influence Functions are a standard tool for attributing predictions to training data in a principled manner and are widely used in applications such as data valuation and fairness. In this work, we present realistic incentives to manipulate influence-based attributions and investigate whether these attributions can be *systematically* altered by an adversary. We show that small systemic perturbations to models can indeed alter influence-based attributions *as desired*. We work on logistic regression models trained on ResNet feature embeddings and standard tabular fairness datasets and provide efficient attacks with backward-friendly implementations. Our work raises questions on the reliability of influence-based attributions in adversarial circumstances.

1 INTRODUCTION

Influence Functions (IFs) are a popular tool for data attribution and have been widely used in many applications such as data valuation (Richardson et al., 2019; Hesse et al., 2023; Sundararajan and Krichene, 2023; Jia et al., 2019), data filtering/subsampling/cleaning (Wu et al., 2022; Wang et al., 2020; Miao et al., 2021; Teso et al., 2021; Meng et al., 2022), fairness (Li and Liu, 2022; Wang et al., 2024a; Sattigeri et al., 2022; Kong et al., 2021; Pang et al., 2024; Chhabra et al., 2023; Chen et al., 2024; Yao and Liu, 2023; Ghosh et al., 2023) and so on. While earlier they were being used for benign debugging, many of these newer applications involve adversarial scenarios where participants have an incentive to manipulate influence scores; for example, in data valuation monetary sum is given to samples according to their contribution to training and therefore model developers have an incentive to superficially lower influence scores for existing data or they might collude with certain data providers and artifically raise their valuation over others. Thus, an understanding of whether and how influence functions can be manipulated is essential to determine their proper usage and for putting guardrails in place.

Literature has studied the manipulation of feature-based attributions (Heo et al., 2019; Anders et al., 2020; Slack et al., 2020) by perturbing *model weights* in a desired fashion. We employ the idea of model manipulation, but for *data-attribution, specifically for influence functions*. To this end, our paper investigates whether any downstream applications of influence function attributions can be *systematically* altered by manipulating model weights according to the manipulator's incentives. Particularly, we focus on two important downstream applications where incentives are meaningful: data valuation, where the adversary has an incentive to change influence scores for monetary gain and fairness, where the adversary wants to manipulate influence scores for reducing the fairness of a downstream model. Note that this is different from existing works which showcase the brittleness of IFs arising from approximation errors and randomness (Basu et al., 2020; Bae et al., 2022; Epifano et al., 2023), our paper studies if a systematic manipulation of IFs while maintaining accuracy is possible. Additionally, our attacks do not rely on the brittleness of IFs and hence might work even under an accurate and stable definition of IFs.

We next define and provide algorithms to carry out two kinds of attacks in this setup: Targeted and Untargeted. Targeted attacks are for the data valuation application and specifically manipulate influence scores for certain target samples. The primary challenge with these attacks is that calculating gradients of influence-based loss objectives is highly computationally infeasible. We address this challenge by proposing a memory-time efficient and backward-friendly algorithm to compute the gradients while using existing PyTorch machinery for

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

implementation. This contribution is of independent technical interest, as the literature has only focused on making forward computation of influence functions feasible, while we study techniques to make the *backward pass* viable. Our algorithm brings down the memory required for one forward + backward pass from not being feasible to run on a 12GB GPU to 7GB for a 206K parameter model and from 8GB to 1.7GB for a 5K model.

Experiments on multiclass logistic regression models trained on ResNet50 features show that our targeted attacks achieve a high success rate, a maximum of 94%, without much accuracy drop across three datasets. One final question that comes to mind is – is it always possible to manipulate the influence scores for any given training sample? Using a theoretical construction, we give an impossibility theorem which states that there exist samples for which the influence score cannot be manipulated irrespective of the model, making this a property of the data rather than the model.

Untargeted attacks are for the fairness application and unlike targeted attacks, manipulate influence scores arbitrarily without targeting specific samples. We find that, surprisingly enough, scaling model weights is a good enough strategy for such attacks without changing model accuracy. In our experiments on standard tabular fairness datasets, we observe that due to influence score manipulation fairness of downstream models is affected a lot, leading to a maximum of 16% difference in fairness with influence manipulation.

Our work exposes the susceptibility of influence-based attributions to manipulation, highlighting the need of careful consideration when used in adversarial contexts. Code is available on our GitHub repository.

2 PRELIMINARIES

Consider a classification task with an input space $\mathcal{X} = \mathbb{R}^d$ and labels in set $\mathcal{Y}$. Let the training set of size n be denoted by $Z = \{z_i\}_{i=1}^n$ where each sample z_i is an input-label pair, $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Let the loss function at a particular sample z and model parameters $\theta \in \Theta$ be denoted by $L(z, \theta)$. Using the loss function and the training set, a model parameterized by $\theta \in \Theta$ is learnt through empirical risk minimization, resulting in the optimal parameters $\theta^\star := \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n L(z_i, \theta)$. The gradient of the loss w.r.t. parameters θ for the minimizer at a sample z is given by $\nabla_\theta L(z, \theta^\star)$. Hessian of the loss for the minimizer is denoted by $H_{\theta^\star} := \frac{1}{n} \sum_{i=1}^n \nabla_\theta^2 L(z_i, \theta^\star)$. For brevity, we call the model parameterized by θ as model θ. Next we give the definition of Influence Functions used in our paper.

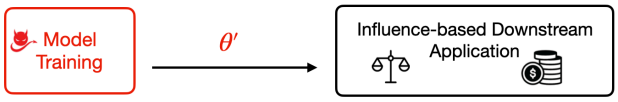

Figure 1: Threat Model. Adversarial manipulation happens in the model training process, where the honest model is manipulated towards a malicious one to achieve desired influence scores, while maintaining test performance.

Definition 1 (Influence Function Koh and Liang (2017))**.** *Assuming that the empirical risk is twice-differentiable and strictly convex in model parameters θ, the influence of a training point z on the loss at a test point z_{test} is given by,*

$$\mathcal{I}_{\theta^\star}(z, z_{\text{test}}) := -\nabla_\theta L(z_{\text{test}}, \theta^\star)^\top H_{\theta^\star}^{-1} \nabla_\theta L(z, \theta^\star) \quad (1)$$

where $\nabla_\theta L(z_{\text{test}}, \theta^\star)$ and $\nabla_\theta L(z, \theta^\star)$ denote the loss gradients at z_{test} and z respectively, while $H_{\theta^\star}^{-1}$ denotes the hessian inverse.

For logistic regression, the influence function has a closed form given by,

$$\begin{aligned}\mathcal{I}_\theta(z, z_{\text{test}}) = -y_{\text{test}} y \cdot \sigma(-y_{\text{test}} \theta^\top x_{\text{test}}) \\ \cdot \sigma(-y\theta^\top x) \cdot x_{\text{test}}^\top H_\theta^{-1} x\end{aligned} \quad (2)$$

where $y \in \{-1, 1\}$ and $\sigma(t) = \frac{1}{1+\exp(-t)}$Koh and Liang (2017).

Given a test set of size m, $Z_{\text{test}} = \{z_{\text{test_i}}\}_{i=1}^m$, we define the overall influence of a training point z on the loss of the test *set* to be the sum of its influence on all test points $z_{\text{test_i}}$ individually, written as

$$\mathcal{I}_{\theta^\star}(z, Z_{\text{test}}) := \sum_{i=1}^m \mathcal{I}_{\theta^\star}(z, z_{\text{test_i}}) \quad (3)$$

3 GENERAL THREAT MODEL

In this section, we give a description of our setup and general threat model. We later instantiate these with two downstream applications, data valuation and fairness, where the objectives and incentives differ. While we ground our discussion on these two applications, the attacks or their slight variations can apply to other applications.

An adversary who has incentives in downstream applications that rely on influence scores, would want influence scores according to his desires. This brings us to the question of how should the adversarial manipulation occur to obtain such desired influences? Prior work on manipulating feature attributions has shown that corrupting the model can successfully manipulate

feature attributions (Heo et al., 2019; Pruthi et al., 2019; Anders et al., 2020; Shahin Shamsabadi et al., 2022). We wonder if the same is true for influence-based attributions leading to the following threat model.

We assume the adversary manipulates the model weights to maliciously change influence scores for some training samples. In addition, the malicious model θ' has similar test accuracy as original honest model $\theta^\star$ while leading to desired influence scores.

One might wonder why the adversary simply doesn't output the desired influence scores? This would be a viable attack only if the manipulation is discreet and cannot be detected; however an auditor can simply ask for model weights and detect this kind of manipulation (same is true for feature-based attributions in previous works).

We further assume the adversary has the training data together with a test set. This can be reasonable because the adversary who can manipulate the (released) model also possibly has access to the data, e.g. when the adversary is the model trainer (Figure 1).

4 DATA VALUATION APPLICATION

The goal of data valuation is to determine the contribution of each training sample to model training and accordingly assign a proportional monetary sum to each. One of the techniques to find this value is through influence functions, by ranking training samples according to their influence scores in a *decreasing order* (Richardson et al., 2019; Hesse et al., 2023; Sundararajan and Krichene, 2023; Jia et al., 2019). A higher (lower) influence ranking implies a more (less) valuable sample, resulting in a higher (lower) monetary sum.

A model trainer with financial incentives would like to artificially reduce the compensation it gives to data vendors or it can collude with certain data vendors to artificially increase their compensation over others. Next we formally specify the threat model corresponding to this setting and the goal of the adversary.

Threat Model. The canonical setting of data valuation consists of multiple data vendors. Each vendor supplies a set of data; the collection of data from all vendors corresponds to the fixed training set. The adversarial model training can change model parameters from θ^* to θ' while maintaining similar test accuracy.

Goal of the adversary. Given a set of target samples $Z_{\text{target}} \subset Z$, the goal of the adversary, without loss of generality, is to push the influence ranking of samples from Z_{target} to top-k or equivalently increase the influence score of samples from Z_{target} beyond the remaining $n-k$ samples, where $k \in \mathbb{N}$. The same techniques apply to push the samples to bottom-k influence rankings.

Given this goal of the adversary, we wonder if this goal can always be achieved or if there exist target samples for which the influence score cannot be moved to top-k rankings? With a theoretical construction of the data, we prove that the latter case is true, formally stating the impossibility theorem next.

Theorem 1. *For a logistic regression family of models and any target influence ranking $k \in \mathbb{N}$, there exists a training set Z_{train}, test set Z_{test} and target sample $z_{\text{target}} \in Z_{\text{train}}$, such that no model in the family can have the target sample z_{target} in top-k influence rankings.*

Since the proof for the theorem (in App. Sec. A.1.3) relies on a specific worst-case construction of the data, the goal of the adversary can still be achieved for realistic data distributions. We next provide algorithms to achieve the adversary's goal for such cases.

4.1 Attack Methodology and Algorithm

We will first describe how we frame the goal of the adversary as an attack and then describe the algorithm to conduct it.

Let us first consider the case where the set of target samples, Z_{target}, has only one element, $Z_{\text{target}} = \{z_{\text{target}}\}$. We call this setting the *Single-Target Attack*. We formulate the adversary's attack as a constrained optimization problem where the objective function, ℓ_{attack}, captures the intent to raise the influence ranking of the target sample to top-k while the constraint function, dist, limits the distance between the original and manipulated model, so that the two models have similar test accuracies. The resulting optimization problem is given as follows, $\min_{\theta':\text{dist}(\theta^\star,\theta')\leq \text{C}} \ell_{\text{attack}}(z_{\text{target}}, Z, Z_{\text{test}}, \theta')$,

where $C \in \mathbb{R}$ is the model manipulation radius and controls the trade-off between the success of the attack and the degree that the model can be perturbed, i.e. the drop of the performance.

Next when the target set consists of multiple target samples, $Z_{\text{target}} = \{z_{\text{target}_1}, z_{\text{target}_2} \cdots z_{\text{target}_q}\}$, we call the attack *Multi-Target Attack*. The adversary's attack can now be formulated as repeated applications of the Single-Target Attack, formally given as,

$$\min_{\theta':\text{dist}(\theta^\star,\theta')\leq \text{C}} \sum_{z_{\text{target}_i} \in Z_{\text{target}}} \ell_{\text{attack}}(z_{\text{target}_i}, Z, Z_{\text{test}}, \theta') \tag{4}$$

The actual objective used for both the attacks is given as, $\ell_{\text{attack}}(\cdot) = -\mathcal{I}_{\theta'}(z_{\text{target}}, Z_{\text{test}}) + \frac{1}{|S_{\theta'}|}\sum_{z \in S_{\theta'}} \mathcal{I}_{\theta'}(z, Z_{\text{test}})$ where $S_{\theta'} \subset Z_{\text{train}}$ contains all training samples z s.t. $\mathcal{I}_{\theta'}(z, Z_{\text{test}}) > \mathcal{I}_{\theta'}(z_{\text{target}}, Z_{\text{test}})$ (see ablation study in Sec. 4.2) to understand why we chose this loss objective). Here the first term maximizes the influence of the target sample z_{target} while the second term minimizes the influence of all samples which are currently more influential than z_{target}. Since this objective is non-convex, the optimization process results in local minima, which might be non-optimal. Therefore to get better results, we run every attack multiple times, starting with random initializations of $\theta^\star$ in a radius C. Our final algorithm is as given in Alg.1.

Efficient Backward Pass for Influence-based Objectives. A natural algorithm to solve complicated optimization problems as our attacks is Gradient Descent, which involves a forward and backward pass. However, for influence-based attack objectives, naive gradient descent is not feasible for either of the passes, mainly due to Hessian-Inverse-Vector Products (HIVPs) in the influence function definition which lead to a polynomial scaling of memory and time requirements w.r.t model parameters. Backward pass on our attack objectives is even harder as it involves *gradients* of influence-based loss objectives, making the attacks too expensive even for linear models trained on top of ResNet50 features used in our experiments where #parameters range from ∼76k-206k.

While literature has studied ways to make the forward computation of influence functions efficient Schioppa et al. (2022); Guo et al. (2020); Koh and Liang (2017); Kwon et al. (2023), not much work has been done on making the *backward pass efficient*. To this end, we propose a simple technique – rewriting the original objective into a backward-friendly form – which renders the gradient computations efficient for influence-based objectives. This allows us to still use gradient descent and other existing machinery in PyTorch (Paszke et al., 2017). Our idea of rewriting the attack objective involves two essential steps : (1) linearizing the objective (2) making the linearized objective backward-friendly in PyTorch, as outlined in Alg.2. This algorithm is of independent technical interest and is generalizable to other use-cases where backward passes through HIVPs are needed. More details about the technique with derivations are given in the Appendix Sec. A.1.1.

4.2 Data Valuation Experiments

In this section, we investigate if the attacks we proposed for data valuation can succeed empirically. Specifically, we ask the following questions : (1) do our influence-based attacks perform better than a non-influence baseline?, (2) what is the behavior of our attacks w.r.t. different parameters such as radius C and target set size? ,(3) what components contribute to the success of our attacks? and (4) lastly, can our attacks transfer to an unknown test set?. In what follows, we first explain our experimental setup and then discuss the main results followed by ablation study.

4.2.1 Setup

Datasets & Models. We use three standard image datasets for experimentation : CIFAR10 (Krizhevsky et al., 2009), Oxford-IIIT Pet (Parkhi et al., 2012) and Caltech-101 (Li et al., 2022). We split the respective test sets into two halves while maintaining the original class ratios for each. The first half is the test set used to optimize influence scores in the attack while the second is used as a pristine set for calculating the accuracy of models and also for transfer experiments discussed later. We pass all images through a pretrained ResNet50 model (He et al., 2016) from PyTorch to obtain feature vectors of size 2048 for and train linear models (θ^*) on top of these features with cross-entropy loss and a learning rate of 0.001.

Attack Setup & Evaluation. The constraint function dist is set to L2 distance. Forward pass for the attacks is using the LiSSA Algorithm Koh and Liang (2017); details including parameters used can be found in App. Sec.A.1.2. For the Single-Target Attack, we randomly pick a training sample (which is not already in the top-k influence rankings) as the target and carry out our attack on it. We repeat this process for 50 samples and report the fraction out of 50 which could be (individually) moved to top-k in influence rankings as the success rate. To carry out the Multi-Target Attack, we randomly pick target sets of different sizes from the training set. The success rate now is the fraction of samples in the target set which could be moved top-k in influence rankings. For many of our results, the success rates are reported under two regimes : (1) the high-accuracy similarity regime where the manipulated and original models are within 3% accuracy difference and (2) the best success rate irrespective of accuracy difference. We optimize every attack from 5 different initializations of $\theta^\star$ within a radius of C and report the runs which eventually lead to the highest success rates.

Baseline: Loss Reweighing Attack. While our attacks are based on influence functions, we propose a non-influence baseline attack for increasing the importance of a training sample : reweigh the training loss, with a high weight on the loss for the target sample. We call this baseline the Loss Reweighing Attack, formally defined as, $\min_{\theta'} \sum_{z \in Z_{\text{train}} \setminus \{z_{\text{target}}\}} L(z; \theta') + \alpha \cdot L(z_{\text{target}}; \theta')$, where L is the model training loss and $\alpha \in \mathbb{R}$ is the weight on the loss of target sample. In-

tuitively, a larger weight α increases the influence of z_{target} on the final model, but results in a lower model accuracy and vice-versa. Directly reweighing the loss as in the baseline led to unstable training, so we instead implemented the baseline with weighted sampling according to weight α in each batch.

For more experimental details, kindly refer to the Appendix Sec. A.1.2. Next we discuss our results.

4.2.2 Main Results

Our Single-Target attack performs better than the Baseline. As demonstrated in Table 1, our influence-based attacks indeed performs better than the baseline – while the baseline has a low success rate across the board, our attack achieves a success rate of 64-88% in the high accuracy regime and 85-94% without accuracy constraints. The baseline is able to achieve a high success rate when ranking k is large, but only with a massive accuracy drop. The fact that our attack did not achieve a 100% success rate highlights that this manipulation problem is non-trivial (thm. 1).

Behavior of our Single-Target attack w.r.t manipulation radius C & training set size. Theoretically, the manipulation radius parameter C in our attack objectives is expected to create a trade-off between the manipulated model's accuracy and the attack success rate. Increasing radius C should result in a higher success rate as the manipulated model is allowed to diverge more from the (optimal) original model but on the other hand its accuracy should drop and vice-versa. We observe this trade-off for all three datasets and different values of ranking k, as shown in Fig.2 (solid lines).

We also anticipate our attack to work better with smaller training sets, as there will be fewer samples competing for top-k rankings. Experimentally, Pet dataset with the smallest training set has the highest success rates (Fig.2, Tab. 1).

Our attacks transfer when influence scores are computed with an unknown test set. When an unknown test set is used to compute influence scores, our attacks perform better as ranking k increases, as shown in Fig.2. This occurs because rank of the target sample, optimized with the original test set, deteriorates with the unknown test set and a larger k increases the likelihood of the target still being in the top-k rankings.

How does our Multi-Target Attack perform with changing target set size and desired ranking k? Intuitively, our attack should perform better when the size of the target set is larger compared to ranking k – this is simply because a larger target set offers more candidates to take the top-k rankings spots, thus increasing the chances of some of them making it to top-k. Our experimental results confirm this intuition; as demonstrated in Fig.3, we observe that (1) for a fixed value of ranking k, a larger target set size leads to a higher success rate; target set size of 100 has the highest success rates for all values of ranking k across the board, and (2) the success rate decreases with increasing value of k for all target set sizes and datasets. These results are for the high-accuracy similarity regime where the original-manipulated model accuracy differ by $\leq 3\%$.

4.2.3 Further Analysis

Ablation Study. Since our attack is a combination of several ideas, we conduct an ablation study to understand the contribution of each idea on the success rate, as reported in Table 2. The different ideas are as follows.

- Maximize the target data's influence: Given a target sample, the simplest idea to move it to top-k influence rankings is to maximize its own influence score, objective written as $\max_{\theta':\text{dist}(\theta^*,\theta')\leq C} \mathcal{I}_{\theta'}(z_{\text{target}}, Z_{\text{test}})$. This attack doesn't achieve high success rates, even without accuracy constraints which could be due to an inherent drawback : this objective doesn't consider other training samples' influence scores.

- + Minimize the influence of samples that are ranked top-k: Instead of just increasing the influence score of the target sample, this objective also lowers the score for the samples currently ranked top-k, given as $\max_{\theta':\text{dist}(\theta^*,\theta')\leq C} \mathcal{I}_{\theta'}(z_{\text{target}}, Z_{\text{test}}) - \frac{1}{K}\sum_{z:\text{rank of } z\leq k} \mathcal{I}_{\theta'}(z, Z_{\text{test}})$. We observe empirically that the optimization procedure of this objective gets stuck in local minima easily.

- + **(Our objective)** Minimize the influence score of all samples whose influence is larger than that of the target sample : This is the final objective used by us and lowers the influence of all training samples which have a higher influence than that of the target sample instead of just the top-k (as in the previous objective), $\min_{\theta':\text{dist}(\theta^*,\theta')\leq C} \frac{1}{|S_{\theta'}|}\sum_{z\in S_{\theta'}} \mathcal{I}_{\theta'}(z, Z_{\text{test}}) - \mathcal{I}_{\theta'}(z_{\text{target}}, Z_{\text{test}})$ where $S_{\theta'} \subseteq Z_{\text{train}}$ has all training samples z s.t. $\mathcal{I}_{\theta'}(z, Z_{\text{test}}) > \mathcal{I}_{\theta'}(z_{\text{target}}, Z_{\text{test}})$. Empirically, we find that this objective function decreases the chance of being stuck at suboptimal solutions and the loss keeps reducing throughout the optimization trajectory resulting in higher success rates.

- + **(Our final attack)** Multiple random initializations. Because the above objective function is non-convex, we find that using multiple random initializations of the honest model within a radius

Table 1: Success Rates of the Baseline vs. our Single-Target Attack for Data Valuation. k is the ranking, as in top-k. $\Delta_{\text{acc}} := \text{TestAcc}(\theta^\star) - \text{TestAcc}(\theta')$ represents drop in test accuracy for manipulated model θ'. Two success rates are reported : (1) when $\Delta_{\text{acc}} \leq 3\%$ (2) the best success rate irrespective of accuracy drop. (%) represents model accuracy. (-) means a model with non-zero success rate could not be found & hence accuracy can't be stated. *Our attack has a significantly higher success rate as compared to the baseline with a much smaller accuracy drop under all settings.*

Dataset (Honest Model $\theta^\star$ Accuracy)		CIFAR10 (89.8%)		Oxford-IIIT Pet (92.2%)		Caltech-101 (94.9%)	
Success Rate		$\Delta_{\text{acc}} \leq 3\%$	Best (Δ_{acc})	$\Delta_{\text{acc}} \leq 3\%$	Best (Δ_{acc})	$\Delta_{\text{acc}} \leq 3\%$	Best (Δ_{acc})
$k = 1$	Baseline	0.00	0.00 (-)	0.00	0.00 (-)	0.00	0.00 (-)
	Our	0.64	0.90 (5.7%)	0.88	0.94 (5.4%)	0.74	0.85 (3.8%)
$k = 300$	Baseline	0.00	0.00 (-)	0.10	1.00 (87.3%)	0.08	0.84 (93.6%)
	Our	0.76	0.90 (5.7%)	0.88	0.94 (5.4%)	0.74	0.85 (3.8%)

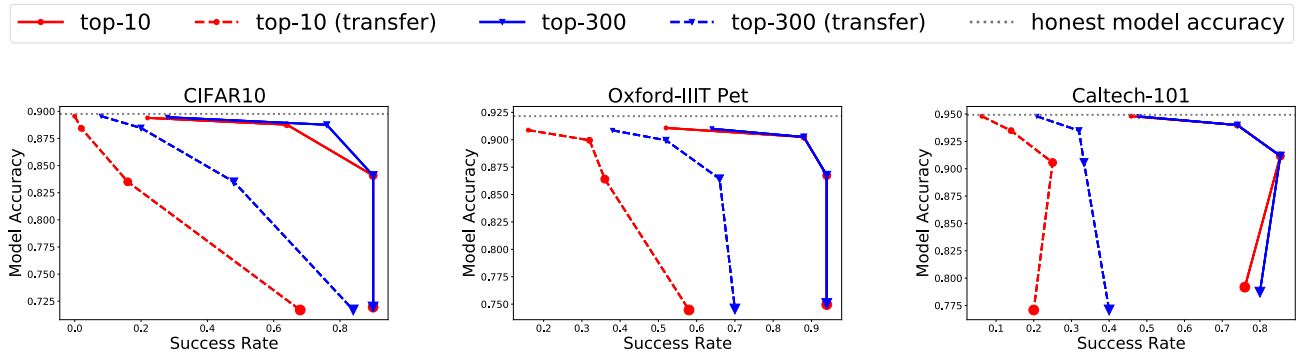

Figure 2: Behavior and Transfer results for Single-Target Attack in the Data Valuation use-case. Value of manipulation radius C (Eq.**??**) increases from left to right in each curve. (1) Behavior on original test set (solid lines) : *As manipulation radius C increases, manipulated model accuracy drops while attack success rate increases.* (2) Transfer on an unknown test set (dashed lines): *Success rate on an unknown test set gets better with increasing values of ranking k.*

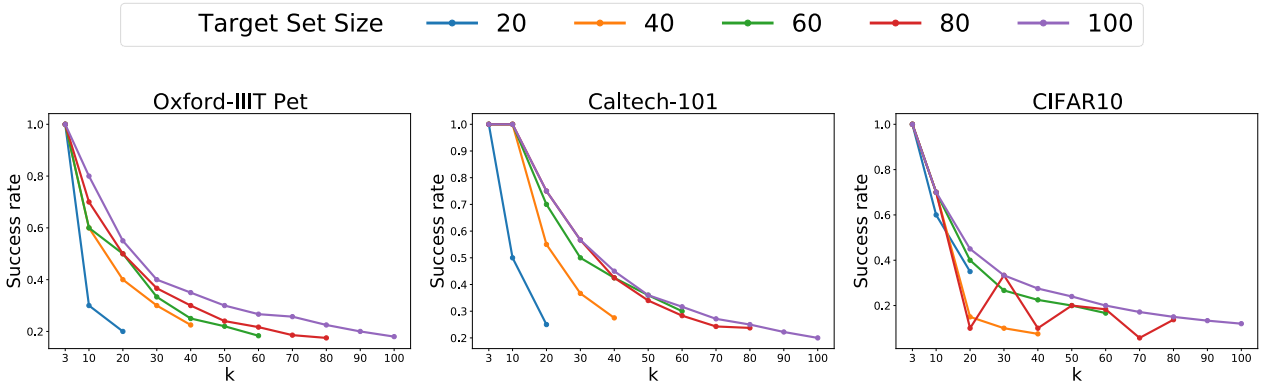

Figure 3: Performance of Multi-Target Attack in the Data Valuation use-case. Results for the high-accuracy regime. *Success Rates are higher when target set size is greater than the desired ranking k.*

C helps to obtain a better solution, especially with a larger value of parameter C, when the search space is bigger. As a result, we observe significant improvement in terms of 'best' success rates (where C can be very large). This is our final attack.

Easy vs. Hard Samples. We find that target samples which rank very high or low in the original influence rankings are easier to push to top-k rankings upon manipulation (or equivalently samples which have a high magnitude of influence either positive or negative). This is because the influence scores of extreme rank samples are more sensitive to model parameters as shown experimentally in App. Fig.5, thus making them more susceptible to influence-based attacks.

Table 2: Ablation study for Single-Target Attack in Data Valuation. Ranking $k := 10$. $\Delta_{\text{acc}} := \text{TestAcc}(\theta^\star) - \text{TestAcc}(\theta')$ represents the drop in test accuracy for a manipulated model. (%) represents model accuracy. Two success rates are reported : (1) when $\Delta_{\text{acc}} \leq 3\%$ and (2) the best success rate irrespective of accuracy drop. *Our final objective with multiple random initializations of original model within radius C leads to highest success rates.*

Dataset (Honest Model $\theta^\star$ Accuracy)	CIFAR10 (89.8%)		Oxford-IIIT Pet (92.2%)		Caltech-101 (94.9%)	
Success Rate	$\Delta_{\text{acc}} \leq 3\%$	Best (Δ_{acc})	$\Delta_{\text{acc}} \leq 3\%$	Best (Δ_{acc})	$\Delta_{\text{acc}} \leq 3\%$	Best (Δ_{acc})
Max. Target Inf.	0.44	0.64 (15.9%)	0.64	0.70 (15.2%)	0.52	0.72 (11.7%)
+ Min. Top-k Inf.	0.60	0.72 (20.0%)	0.74	0.74 (1.6%)	0.68	0.68 (3.6%)
+ Min. Higher-Rank Inf.	0.60	0.80 (6.0%)	0.88	0.88 (2.1%)	0.74	0.77 (8.3%)
+ Multiple Rand Init. (**Ours**)	**0.64**	**0.90** (5.5%)	**0.88**	**0.94** (5.3%)	**0.74**	**0.85** (7.1%)

5 FAIRNESS APPLICATION

Recently, a lot of studies have used influence functions in different ways to achieve fair models (Li and Liu, 2022; Wang et al., 2024a; Sattigeri et al., 2022; Kong et al., 2021; Pang et al., 2024; Chhabra et al., 2023; Chen et al., 2024; Yao and Liu, 2023). In our paper, we focus on the study by Li and Liu (2022) as they use the same definition of influence functions as us. The suggested approach in Li and Liu (2022) to achieve a fair model is by *reweighing training data* based on influence scores for a *base model* and then using this reweighed data to train a new downstream model from scratch. This downstream model is expected to have high fairness as a result of the reweighing. For a pictorial representation of this process, see App. Fig. 6. The weights for training data are found by solving an optimization problem which places influence functions in the constraints. Optimization problem details are in App. Sec.A.2.

Ultimately, weights of the training data determine the fairness of the downstream model. Since these weights are derived from influence scores, manipulating the influence scores can alter the fairness of the downstream model. As a result, a malicious entity who wants to spread unfairness is incentivized to manipulate influence scores.

Threat Model. Training and test sets are fixed. The base model is trained by entity X and put out publicly along with training set. Entity Y wants to train a model on this dataset but wants higher fairness, they use an IF-based methodology to achieve this higher-fairness model. Here, X is the adversary and tampers the *base model* parameters from θ^* to θ' to manipulate IF scores so that the model achieved by entity Y has a lower downstream fairness.

Goal of the adversary. Fairness of the final downstream model is measured with a fairness metric. The concrete goal of the adversary is to make the value of this fairness metric worse for the downstream model than what could have been achieved without manipulation.

Attack Methodology. Since the goal of the adversary in this case is not tied to specific target samples, we propose an untargeted attack for the adversary. Our attack is deceptively simple – scale the base model $\theta^\star$ by a constant $\lambda > 0$. The malicious base model output by the model trainer is now $\theta' = \lambda \cdot \theta^\star$, instead of $\theta^\star$. Note that for logistic regression the malicious and original base model are indistinguishable since scaling with a positive constant maintains the sign of the predictions, leading to the same accuracy.

5.1 Fairness Manipulation Experiments

Setup. We investigate experimentally if our scaling attack can be successful in reducing fairness of the final downstream model. We use three standard fairness benchmarks : Adult Credit Becker and Kohavi (1996), German Credit Hofmann (1994) and Compas Angwin et al. (2022). For details about these datasets, see Appendix Table 3. We train L2-regularized logistic regression models for these datasets which act as base models for the training data reweighing pipeline. To carry out our fairness manipulation attack, we scale the models weights with different scaling coefficients and feed these scaled models to the reweighing pipeline. The reweighed training data is then used to train a final downstream model, which is expected to be fair. Parameter values for L2 regularization and the reweighing are same as in Li and Liu (2022), mentioned in App. Tables 4, 3.

Evaluation. We evaluate the fairness of the final downstream model with a popular fairness metric, called demographic parity (DP) (results for 'Equality of Opportunity' Hardt et al. (2016) fairness metric and definition of DP can be found in Appendix Sec. A.2). A higher value of DP gap implies less fairness and vice-versa. We call the attack a success if the final downstream model post-attack is less fair (higher DP gap) than the final model achieved without attacking. Additionally, we expect the test accuracies of the downstream model with and without attack to be similar to keep the manipulation discreet. Note that this is different from our threat model's indistinguishability constraint, which applies to the base model, not to the

downstream model.

Results. As can be seen from our results in Fig.4, the scaling attack works surprisingly well across all datasets – downstream models achieved after our attack are considerably less fair (higher DP gap) than the models without attack, achieving a maximum difference of 16% in the DP gap. Simultaneously, downstream models post-attack maintain similar test accuracies to downstream models without attack. Since the process to achieve the downstream model involves a lot of steps, including solving a non-convex optimization problem to find training data weights and then retraining a model, we sometimes do not see a smooth monotonic trend in fairness metric values w.r.t. scaling coefficients. However, this does not matter much from the attacker's perspective as all the attacker needs is *one* scaling coefficient which meets the attack success criteria.

6 DISCUSSION

Susceptibility and Defense. The susceptibility of influence functions to our attacks can come from the fact that there can exist models that behave very similarly (Rashomon Effect Rudin et al. (2024)) but have different influential samples up to an extent.Some plausible ways to defend against the attacks are (1) providing cryptographic proofs of honest model training using Zero-Knowledge Proofs Sun et al. (2023); Abbaszadeh et al. (2024); Yadav et al. (2024) (2) to check if the model is atleast a local minima or not, since IFs assume that the model is an optimal solution.

Other Attribution Methods. Given a test point, attribution methods such as TRAK Park et al. (2023) and Datamodels Ilyas et al. (2022) take the training data as input and output the attribution score for each training point, independent of any specific model. On the other hand, IF-based attribution is tied to a specific model. Therefore the threat models for these attribution methods are different. For methods such as TRAK and Datamodels, we believe the more feasible attack is to manipulate the data (while being discreet) rather than manipulating the model. Note that data manipulation can also be another way of manipulating IFs and could be a good future research directionMahmood et al. (2025); Wang et al. (2024b).

7 RELATED WORK

Fragility of Influence Functions. Influence functions proposed in Koh and Liang (2017) are an approximation to the effect of upweighting a training sample on the loss at a test point. This approximation error can be large as shown by (Basu et al., 2020; Bae et al., 2022; Epifano et al., 2023), making influence functions fragile especially for deep learning models. Our work is *orthogonal* to this line of work as we study the robustness of influence functions w.r.t. *model parameters* instead of approximation error of influence functions w.r.t. the true influence. Additionally we look at systematic manipulation of IFs as desired, rather than brittleness arising from randomness, and approximation errors. Our attacks do not rely on the brittleness of IFs and hence might work even under an accurate and stable definition of IFs.

Model Manipulation in the Threat Model. Manipulating models to execute attacks is a prevalent theme in the literature. Slack et al. (2020) use model manipulations to corrupt feature attributions in tabular data while (Heo et al., 2019; Anders et al., 2020) do so in vision models. Pruthi et al. (2019) corrupt attention-based explanations for language models while maintaining model accuracy. Shahin Shamsabadi et al. (2022) show that it is possible to corrupt a fairness metric by manipulating an interpretable surrogate of a black-box model while maintaining empirical performance of the surrogate. Similar to these, our threat model also allows the adversary to manipulate models while maintaining the test accuracy. However, our adversarial goal is to corrupt influence-based attributions.

Data Manipulation Attack on Explanations. (Ghorbani et al., 2019; Alvarez Melis and Jaakkola, 2018; Zhang et al., 2020; Dombrowski et al., 2019; Kindermans et al., 2019) have studied how data can be manipulated to corrupt feature attributions. On the contrary, firstly, we keep data fixed and manipulate the model and secondly, we work with data attributions rather than feature attributions.

8 CONCLUSION & FUTURE WORK

While past work has mostly focused on feature attributions, in this paper we exhibit realistic incentives to manipulate data attributions. Motivated by the incentives, we propose attacks to manipulate outputs from a popular data attribution tool – Influence Functions. We demonstrate the success of our attacks experimentally on multiclass logistic regression models on ResNet features and standard tabular fairness datasets. Our work lays bare the vulnerablility of influence-based attributions to manipulation and serves as a cautionary tale when using them in adversarial circumstances.

Acknowledgements

This work was supported by grants NSF CIF-2402817, NSF CNS-1804829, SaTC-2241100, CCF2217058, ARO-MURI W911NF2110317 and ONR under N00014-24-1-2304.

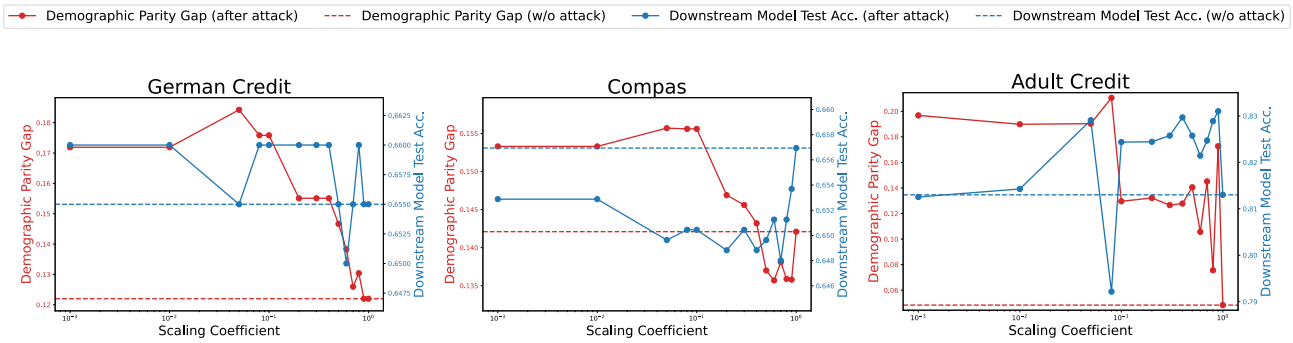

Figure 4: Scaling attack for the Fairness use-case. *Demographic Parity Gap of post-attack downstream models is higher than that of those w/o attack while test accuracies are comparable. This implies that post-attack downstream models are less fair than those w/o attacks.* Scaling coefficients in log scale.

References

Abbaszadeh, K., Pappas, C., Katz, J., and Papadopoulos, D. (2024). Zero-knowledge proofs of training for deep neural networks. *Cryptology ePrint Archive*.

Alvarez Melis, D. and Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31.

Anders, C., Pasliev, P., Dombrowski, A.-K., Müller, K.-R., and Kessel, P. (2020). Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pages 314–323. PMLR.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2022). Machine bias. In *Ethics of data and analytics*, pages 254–264. Auerbach Publications.

Bae, J., Ng, N., Lo, A., Ghassemi, M., and Grosse, R. B. (2022). If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967.

Basu, S., Pope, P., and Feizi, S. (2020). Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*.

Becker, B. and Kohavi, R. (1996). Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

Chen, R., Yang, J., Xiong, H., Bai, J., Hu, T., Hao, J., Feng, Y., Zhou, J. T., Wu, J., and Liu, Z. (2024). Fast model debias with machine unlearning. *Advances in Neural Information Processing Systems*, 36.

Chhabra, A., Li, P., Mohapatra, P., and Liu, H. (2023). "what data benefits my classifier?" enhancing model performance and interpretability through influence-based data selection.

Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., and Kessel, P. (2019). Explanations can be manipulated and geometry is to blame. *Advances in neural information processing systems*, 32.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226.

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

Ghorbani, A., Abid, A., and Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688.

Ghosh, B., Basu, D., and Meel, K. S. (2023). "how biased are your features?": Computing fairness influence functions with global sensitivity analysis. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 138–148.

Guo, H., Rajani, N. F., Hase, P., Bansal, M., and Xiong, C. (2020). Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*.

Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heo, J., Joo, S., and Moon, T. (2019). Fooling neural network interpretations via adversarial model manipulation. *Advances in neural information processing systems*, 32.

Hesse, J., Boldini, D., and Sieber, S. (2023). Data valuation: A novel approach for analyzing high throughput screen data using machine learning.

Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. (2022). Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*.

Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., and Spanos, C. J. (2019). Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR.

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2019). The (un) reliability of saliency methods. *Explainable AI: Interpreting, explaining and visualizing deep learning*, pages 267–280.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.

Kong, S., Shen, Y., and Huang, L. (2021). Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*.

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

Kwon, Y., Wu, E., Wu, K., and Zou, J. (2023). Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. *arXiv preprint arXiv:2310.00902*.

Li, F.-F., Andreeto, M., Ranzato, M., and Perona, P. (2022). Caltech 101.

Li, P. and Liu, H. (2022). Achieving fairness at no utility cost via data reweighing with influence. In *International conference on machine learning*, pages 12917–12930. PMLR.

Mahmood, S. H. A., Yin, M., and Khanna, R. (2025). On the support vector effect in dnns: Rethinking data selection and attribution. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1020–1031.

Meng, X., Yang, Y., Liu, X., and Jiang, N. (2022). Active forgetting via influence estimation for neural networks. *International Journal of Intelligent Systems*, 37(11):9080–9107.

Miao, X., Wu, Y., Chen, L., Gao, Y., Wang, J., and Yin, J. (2021). Efficient and effective data imputation with influence functions. *Proceedings of the VLDB Endowment*, 15(3):624–632.

Pang, J., Wang, J., Zhu, Z., Yao, Y., Qian, C., and Liu, Y. (2024). Fair classifiers without fair training: An influence-guided data sampling approach. *arXiv preprint arXiv:2402.12789*.

Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. (2023). Trak: Attributing model behavior at scale. *arXiv preprint arXiv:2303.14186*.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. (2012). Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Pruthi, D., Gupta, M., Dhingra, B., Neubig, G., and Lipton, Z. C. (2019). Learning to deceive with attention-based explanations. *arXiv preprint arXiv:1909.07913*.

Richardson, A., Filos-Ratsikas, A., and Faltings, B. (2019). Rewarding high-quality data via influence functions. *arXiv preprint arXiv:1908.11598*.

Rudin, C., Zhong, C., Semenova, L., Seltzer, M., Parr, R., Liu, J., Katta, S., Donnelly, J., Chen, H., and Boner, Z. (2024). Amazing things come from having many good models. *arXiv preprint arXiv:2407.04846*.

Sattigeri, P., Ghosh, S., Padhi, I., Dognin, P., and Varshney, K. R. (2022). Fair infinitesimal jackknife: Mitigating the influence of biased training data points without refitting. *Advances in Neural Information Processing Systems*, 35:35894–35906.

Schioppa, A., Zablotskaia, P., Vilar, D., and Sokolov, A. (2022). Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186.

Shahin Shamsabadi, A., Yaghini, M., Dullerud, N., Wyllie, S., Aïvodji, U., Alaagib, A., Gambs, S., and Papernot, N. (2022). Washing the unwashable: On the (im) possibility of fairwashing detection. *Advances in Neural Information Processing Systems*, 35:14170–14182.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186.

Sun, H., Bai, T., Li, J., and Zhang, H. (2023). Zkdl: Efficient zero-knowledge proofs of deep learning training. *Cryptology ePrint Archive*.

Sundararajan, M. and Krichene, W. (2023). Inflow, outflow, and reciprocity in machine learning. In *International Conference on Machine Learning*, pages 33195–33208. PMLR.

Teso, S., Bontempelli, A., Giunchiglia, F., and Passerini, A. (2021). Interactive label cleaning with example-based explanations. *Advances in Neural Information Processing Systems*, 34:12966–12977.

Wang, H., Wu, Z., and He, J. (2024a). Fairif: Boosting fairness in deep learning via influence functions with validation set sensitive attributes. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 721–730.

Wang, X., Hu, P., Deng, J., and Ma, J. W. (2024b). Adversarial attacks on data attribution. *arXiv preprint arXiv:2409.05657*.

Wang, Z., Zhu, H., Dong, Z., He, X., and Huang, S.-L. (2020). Less is better: Unweighted data subsampling via influence function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6340–6347.

Wu, G., Hashemi, M., and Srinivasa, C. (2022). Puma: Performance unchanged model augmentation for training data removal. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 8675–8682.

Yadav, C., Chowdhury, A. R., Boneh, D., and Chaudhuri, K. (2024). Fairproof: Confidential and certifiable fairness for neural networks. *arXiv preprint arXiv:2402.12572*.

Yao, Y. and Liu, Y. (2023). Understanding unfairness via training concept influence. *arXiv preprint arXiv:2306.17828*.

Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., and Wang, T. (2020). Interpretable deep learning under fire. In *29th {USENIX} security symposium ({USENIX} security 20)*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No] We do an empirical investigation.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No] We will make it public upon acceptance.
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] We will make it public upon acceptance.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [No] We will make it public upon acceptance.
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

A Appendix

A.1 Data Manipulation Attack Details

A.1.1 Efficient Backward Pass Algorithm

Our idea of rewriting the attack objective involves two essential steps : (1) linearizing the objective (2) making the linearized objective backward-friendly in PyTorch.

Linearize the attack objective: Generally the attack objective can be a non-linear combination of influence functions over different training samples, which makes the backward pass inefficient. Therefore we first transform the given objective into a linear combination of influence functions, $\hat{\ell}_{\text{attack}}((\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z)) := u^\top(\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z)$ for some vector $u \in \mathbb{R}^n$, such that, the objective and gradient values are the same, $\hat{\ell}_{\text{attack}}(\cdot) = \ell_{\text{attack}}(\cdot)$ and $\nabla_\theta \hat{\ell}_{\text{attack}}(\cdot) = \nabla_\theta \ell_{\text{attack}}(\cdot)$. Observe that from chain rule $\nabla_\theta \ell_{\text{attack}}(\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z) = \sum_{z \in Z} \frac{\partial \ell_{\text{attack}}}{\partial \mathcal{I}_\theta(z, Z_{\text{test}})} \cdot \nabla_\theta \mathcal{I}_\theta(z, Z_{\text{test}})$. Therefore, we can set u as $\left(\frac{\partial \ell_{\text{attack}}}{\partial \mathcal{I}_\theta(z, Z_{\text{test}})} : z \in Z\right)$ while meeting the equal objective and gradient value requirement. Influence scores in this vector are computed using an efficient forward pass algorithm, given in Alg. 3.

Get a PyTorch backward-friendly attack objective: Simply expanding our linearized attack objective gives, $\hat{\ell}_{\text{attack}}(\cdot) = v_{\theta,1}^\top H_\theta^{-1} v_{\theta,2}$ where $v_{\theta,1} = (-\nabla_\theta \sum_{i=1}^m L(z_{\text{test_i}}, \theta))$ and $v_{\theta,2} = \left(\nabla_\theta \sum_{z \in Z} u_z \cdot L(z, \theta)\right)$. Gradient computations for this objective will have to go through HIVPs, which is highly inefficient. Therefore we next convert the linearized objective into one which does not involve HIVPs, again such that the objective and gradient values are same as the original objective.

Using chain rule, the gradient of the expanded linearized attack objective can be written as $\nabla_\theta \hat{\ell}_{\text{attack}}(\cdot) = (\nabla_\theta v_{\theta,1})^\top u_2 + u_1^\top (\nabla_\theta v_{\theta,2}) - u_1^\top (\nabla_\theta H_\theta) u_2$ where $u_1 = H_\theta^{-1} v_{\theta,1}$ and $u_2 = H_\theta^{-1} v_{\theta,2}$. PyTorch supports the gradient computation for functions of gradient, making $v_{\theta,1}$ and $v_{\theta,2}$ backward-friendly. PyTorch also calculates gradients for functions of hessian vector products *implicitly*, which leads to efficiency. Additionally, we can precompute u_1, u_2 and freeze them.

As a result, our final backward-friendly objective function is efficient and backward-friendly with PyTorch and is given as, $\bar{\ell}_{\text{attack}}(\theta) = v_{\theta,1}^\top u_2 + u_1^\top v_{\theta,2} - u_1^\top H_\theta u_2$. The algorithm for computing our backward-friendly objective $\bar{\ell}_{\text{attack}}$ is elucidated in Alg. 2.

Derivation for expanding the linearized objective:

$$\begin{aligned}
\hat{\ell}_{\text{attack}}((\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z)) &= u^\top (\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z) \\
&= \sum_{z \in Z} u_z \cdot \mathcal{I}_\theta(z, Z_{\text{test}}) \\
&= \sum_{z \in Z} u_z \cdot \sum_{i=1}^{m} \mathcal{I}_\theta(z_{\text{test_i}}, z) \\
&= \sum_{z \in Z} u_z \cdot \sum_{i=1}^{m} -\nabla_\theta L\left(z_{\text{test,i}}, \theta\right)^\top H_\theta^{-1} \nabla_\theta L(z, \theta) \\
&= \left(-\nabla_\theta \sum_{i=1}^{m} L\left(z_{\text{test,i}}, \theta\right)\right)^\top H_\theta^{-1} \left(\nabla_\theta \sum_{z \in Z} u_z \cdot L(z, \theta)\right) \\
&= v_{\theta,1}^\top H_\theta^{-1} v_{\theta,2}
\end{aligned}$$

where $v_{\theta,1} = (-\nabla_\theta \sum_{i=1}^{m} L\left(z_{\text{test,i}}, \theta\right))$ and $v_{\theta,2} = \left(\nabla_\theta \sum_{z \in Z} u_z \cdot L(z, \theta)\right)$.

Chain Rule for gradient of expanded linearized objective:

$$\begin{aligned}
\nabla_\theta \hat{\ell}_{\text{attack}}((\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z)) &= \nabla_\theta v_{\theta,1}^\top H_\theta^{-1} v_{\theta,2} \\
&= \left(\nabla_\theta v_{\theta,1}\right)^\top \cdot H_\theta^{-1} v_{\theta,2} + v_{\theta,1}^\top H_\theta^{-1} \left(\nabla_\theta v_{\theta,2}\right) - v_{\theta,1}^\top H_\theta^{-1} \left(\nabla_\theta H_\theta\right) H_\theta^{-1} v_{\theta,2} \\
&= \left(\nabla_\theta v_{\theta,1}\right)^\top u_2 + u_1^\top \left(\nabla_\theta v_{\theta,2}\right) - u_1^\top \left(\nabla_\theta H_\theta\right) u_2
\end{aligned}$$

where $u_1 = H_\theta^{-1} v_{\theta,1}$ and $u_2 = H_\theta^{-1} v_{\theta,2}$.

Algorithm 1 Gradient-based optimization for Attack Loss ℓ_{attack}

Input: Parameters $\theta^\star$, Radius C, train set Z, test set Z_{test}, loss L, attack objective ℓ_{attack}, gradient-based optimizer *Opt*, distance function dist

Output: θ'

1: Set $\theta_0 :=$ Randomly chosen model parameters from a ball of radius C centered around $\theta^\star$, $\mathcal{B}(\theta^\star, C)$ where distance is calculated using dist
2: **for** $t = 1$ **to** T **do**
3: $\bar{\ell}_{\text{attack}}(\theta_{t-1}) =$ Get_Backward_Friendly_Attack_Objective(θ_{t-1}, Z, Z_{test}, L, ℓ_{attack})
4: Compute $\nabla_\theta \bar{\ell}_{\text{attack}}(\theta_{t-1})$ through $\bar{\ell}_{\text{attack}}(\theta_{t-1})$.Backward() in PyTorch
5: Update $\theta_{t-1} \to \theta_t$ by the given gradient-based optimizer *Opt*
6: If $\theta_t \notin \mathcal{B}(\theta^\star, C)$, clip θ_t to lie within $\mathcal{B}(\theta^\star, C)$ where distance is calculated using dist
7: **end for**
8: **Return:** $\theta' := \theta_T$

Algorithm 2 Get_Backward_Friendly_Attack_Objective

Input: Model Parameters θ, Train Set Z, Test Set Z_{test}, Loss L, Original Attack Objective ℓ_{attack}

Output: Backward-Friendly Attack Objective $\bar{\ell}_{\text{attack}}(\theta)$

1: Compute $(\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z)$ from Appendix Alg. 3
2: Compute $u := \left(\frac{\partial \ell_{\text{attack}}}{\partial \mathcal{I}_\theta(z, Z_{\text{test}})} : z \in Z\right)$
3: Compute $v_{\theta,1} := (-\nabla_\theta \sum_{i=1}^{m} L\left(z_{\text{test_i}}, \theta\right))$ and $v_{\theta,2} := \left(\nabla_\theta \sum_{z \in Z} u_z \cdot L(z, \theta)\right)$
4: Compute and freeze $u_1 := H_\theta^{-1} v_{\theta,1}$ and $u_2 := H_\theta^{-1} v_{\theta,2}$
5: Compute $\bar{\ell}_{\text{attack}}(\theta) := v_{\theta,1}^\top u_1 + u_2^\top v_{\theta,2} - u_1^\top H_\theta u_2$
6: **Return:** $\bar{\ell}_{\text{attack}}(\theta)$

When using the original influence-based objective naively, it was not possible to even do one backward pass due to memory constraints. This algorithm brings down the memory required for one forward + backward pass from

Algorithm 3 ForwardOnlyInf

Input: Parameters θ, train set Z, test set Z_{test}, loss L
Output:$(\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z)$
1: Compute $L\left(Z_{\text{test}}, \theta\right) := \nabla_\theta \sum_{i=1}^{m} L\left(z_{\text{test_i}}, \theta\right)$
2: Compute $s_{\text{test}} := H_\theta^{-1} L\left(Z_{\text{test}}, \theta\right)$ by the hessian-inverse-vector product in Koh and Liang (2017).
3: $\forall z \in Z$, compute $\mathcal{I}_\theta(z, Z_{\text{test}}) := s_{\text{test}}^\top \nabla_\theta L\left(z, \theta\right)$
4: **Return:** $(\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z)$

not being feasible to run on a 12GB GPU to 7GB for a 206K parameter model and from 8GB to 1.7GB for a 5K model.

A.1.2 Experimental details

Dataset Details. CIFAR10 (Krizhevsky et al., 2009) has a training/test set size of 50000/10000 with 10 output classes, Oxford-IIIT Pet (Parkhi et al., 2012) has a training/test set size of 3680/3669 with 37 output classes and Caltech-101 (Li et al., 2022) has a training/test set size of 6941/1736 with 101 output classes.

Forward pass Details. We use the LiSSA implementation given at `https://github.com/nimarb/pytorch_influence_functions` with recursion depth=1e6, scale=25 and damping factor= 0.01.

Attack Details. To optimize our attack objective, we use algorithm Alg. 1 for computing gradients with Adam as the optimizer Kingma and Ba (2014). We use two learning rates $\{0.01, 0.1\}$ and 100 steps of updates. We optimize every attack from 5 different initializations. We run our attacks with multiple values of the constraint radius $C = \{0.05, 0.1, 0.2, 0.5\}$. For each regime, the reported number is the highest we could obtain with different values of constants C or α.

Baseline Details. For training a model under the baseline attack we choose Adam as the optimizer, set the batch size as 256 and update for 1400 steps. We ran the baseline for different weights α with a logarithmic scaling from 10 to 1e18. We use weighted sampling instead of uniform to account for the weight on the target sample.

All the experiments were run on internal clusters with NVIDIA GeForce RTX 3080 GPUs.

A.1.3 Proof of Impossibility Theorem (Theorem 1)

Proof of Theorem 1]:
We first introduce a construction of 2-class classification dataset $Z_{\text{train}}, Z_{\text{test}} \subseteq \mathbb{R}^d \times \{1, -1\}$ and show under this construction and a logistic regression model θ, there exists a target data $z_{\text{target}} \in Z_{\text{train}}$ such that no matter how we manipulate this linear model θ, i.e. $\forall \theta \in \mathbb{R}^d$, the rank of $\mathcal{I}_\theta(z_{\text{target}}, Z_{\text{test}})$ among $(\mathcal{I}_\theta(z, Z_{\text{test}}) : z \in Z_{\text{train}})$ *cannot* reach top-1.

Denote $z_i \in \mathbb{R}^d$ as a one-hot vector which has all zeros except 1 at the ith dimension. We construct Z_{test} as $\{(\mathbf{e}_1, 1)\}$, and construct $Z_{\text{train}} = \{(z_i, y_i) | i = 2, \cdots, d\} \cup \{(\mathbf{e}_1, 1), (-\mathbf{e}_1, 1)\}$ where y_i can be arbitrarily selected from $\{1, -1\}$. By choosing $z_{\text{target}} = (\mathbf{e}_1, 1) \in Z_{\text{train}}$ and $z_{\text{bar}} = (-\mathbf{e}_1, 1) \in Z_{\text{train}}$, next we are going to prove $\mathcal{I}_\theta(z_{\text{target}}, Z_{\text{test}}) < \mathcal{I}_\theta(z_{\text{bar}}, Z_{\text{test}})$ for any $\theta \in \mathbb{R}^d$, which indicates $\mathcal{I}_\theta(z_{\text{target}}, Z_{\text{test}})$ among $(\mathcal{I}_\theta(z_{\text{target}}, Z_{\text{test}}) : z \in Z_{\text{train}})$ *cannot* reach top-1.

We adopt the computation of influence function on logistic regression from the original influence function paper (Koh and Liang, 2017). Denote $r(z_i, \theta) = \sigma(z_i^\top \theta) \cdot \sigma(-z_i^\top \theta)$ where $\sigma(t) = \frac{1}{1+\exp(-t)} > 0$. Then the Hessian H_θ of the logistic regression training loss $\frac{1}{d+1} \sum_{z \in Z_{\text{train}}} L(z, \theta)$ is $H_\theta = \frac{1}{d+1} \left(2r(\mathbf{e}_1, \theta) \mathbf{e}_1 \mathbf{e}_1^\top + \sum_{i=2}^{d} r(z_i, \theta) z_i z_i^\top \right)$. Then we can calculate the influence function for z_{target} and z_{bar}:

$$\mathcal{I}_\theta(z_{\text{target}}, Z_{\text{test}}) = -\sigma(-\theta^\top \mathbf{e}_1) \cdot \sigma(\theta^\top \mathbf{e}_1) \cdot \mathbf{e}_1^\top H_\theta^{-1} \mathbf{e}_1 = -\sigma(-\theta^\top \mathbf{e}_1) \cdot \sigma(\theta^\top \mathbf{e}_1) \cdot \frac{d+1}{r(\mathbf{e}_1, \theta)} < 0,$$

$$\mathcal{I}_\theta(z_{\text{bar}}, Z_{\text{test}}) = -\sigma(-\theta^\top \mathbf{e}_1) \cdot \sigma(\theta^\top \mathbf{e}_1) \mathbf{e}_1^\top H_\theta^{-1} (-\mathbf{e}_1) = \sigma(-\theta^\top \mathbf{e}_1) \cdot \sigma(\theta^\top \mathbf{e}_1) \cdot \frac{d+1}{r(\mathbf{e}_1, \theta)} > 0.$$

This complete the proof: $\forall\theta$, $\mathcal{I}_\theta(z_{\text{target}}, Z_{\text{test}}) < 0 < \mathcal{I}_\theta(z_{\text{bar}}, Z_{\text{test}})$ and therefore z_{target} can never achieve top-1.

The above construction can be generalized to top-K for any $K > 1$: we can construct Z_{test} as $\{(\mathbf{e}_1, 1)\}$, and construct $Z_{\text{train}} = \{(z_i, y_i) | i = 2, \cdots, d\} \cup \{(\mathbf{e}_1, 1), K \text{ duplicated } (-\mathbf{e}_1, 1)\}$ where y_i can be arbitrarily selected from $\{1, -1\}$. Then similar to the proof above, by choosing $z_{\text{target}} = (\mathbf{e}_1, 1) \in Z_{\text{train}}$ and K duplicated training point in the training set $z^k_{\text{bar}} = (-\mathbf{e}_1, 1), k \in [K]$, we can prove $\forall\theta$, $\mathcal{I}_\theta(z_{\text{target}}, Z_{\text{test}}) < 0$ and $\forall\theta, k \in [K]$, $\mathcal{I}_\theta(z^k_{\text{bar}}, Z_{\text{test}}) > 0$. Consequently, $\forall\theta$, z_{target} will not achieve top-K.

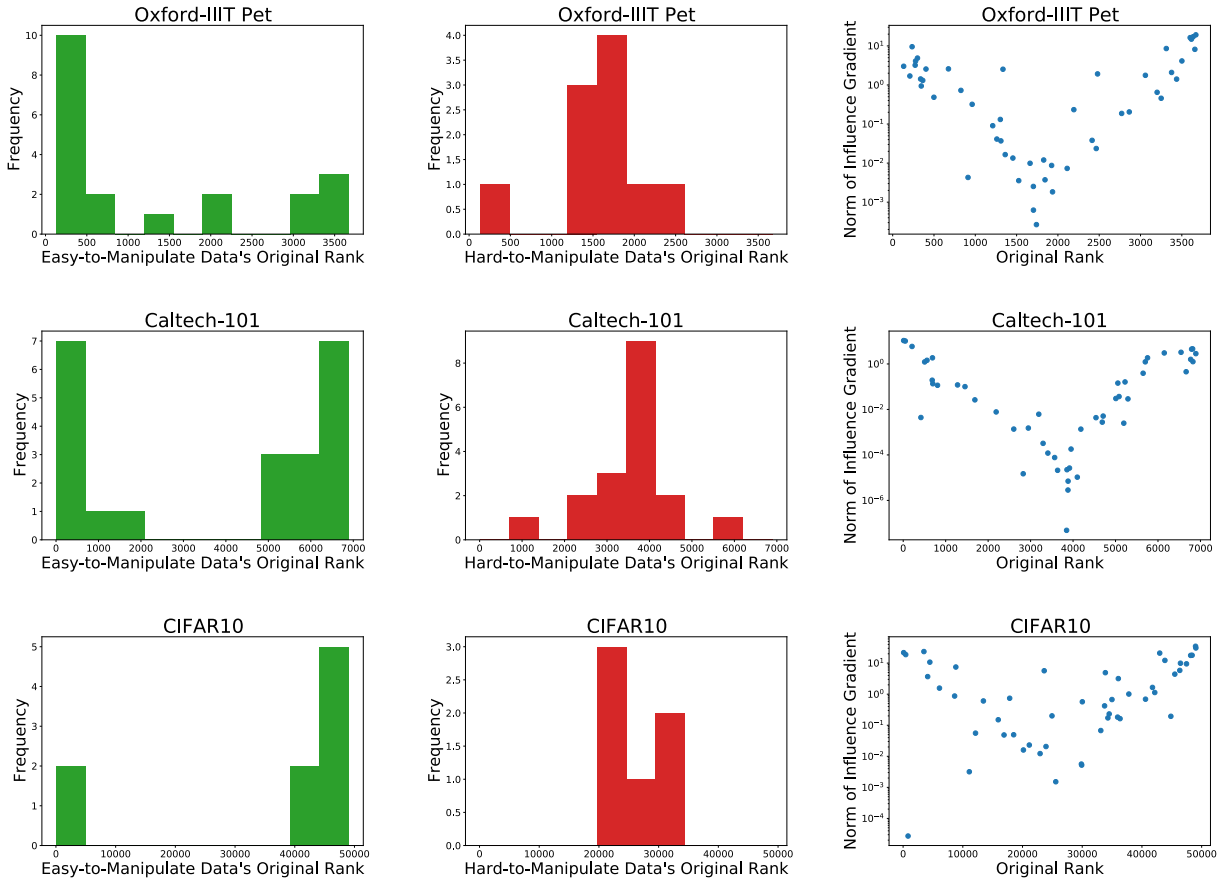

Figure 5: Histograms for original ranks of easy-to-manipulate samples, Histograms for original ranks of hard-to-manipulate samples, Scatterplots for influence gradient norm vs. original ranks of the 50 random target samples. Ranking $k := 1$. *Easy-to-manipulate samples have extreme original influence ranks (large positive or negative) as the samples with the extreme rankings also have higher influence gradient norms, where the gradient is taken w.r.t. model parameters.*

A.2 Fairness Manipulation Attack Details

Optimization Problem for Reweighing Training Data as proposed by Li and Liu (2022) is given as follows,

$$\begin{array}{ll} \text{minimize} & \sum_i w_i \\ \text{subject to} & \sum_i w_i \mathcal{I}_{\text{fair}}\ (z_i) = -f^{\mathcal{V}}_{\text{fair}} \\ & \sum_i w_i \mathcal{I}_{\text{util}}\ (z_i) \leq 0 \\ & w_i \in [0, 1] \end{array} \tag{5}$$

where w_i refers to the weight of the ith training sample, z_i refers to the ith training sample, $\mathcal{I}_{\text{util}}$ refers to our influence function, $\mathcal{I}_{\text{fair}}$ refers to some fairness influence function and $f^{\mathcal{V}}_{\text{fair}}$ corresponds to a differentiable fairness metric. In our threat model, the adversary manipulates the base model, which changes the influence scores $\mathcal{I}_{\text{util}}$.

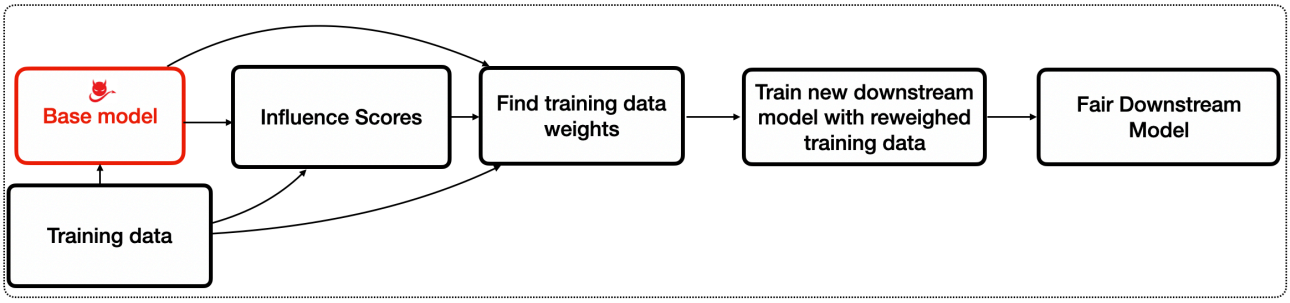

Figure 6: Overview of the process to achieve a fair model, as proposed by Li and Liu (2022). Our adversary, the model trainer manipulates the base model used to calculate influence scores.

An advanced version of the above optimization problem using additional parameters (β, γ) which lead to various tradeoffs is given as,

$$\begin{array}{ll} \text{minimize} & \sum_i w_i \\ \text{subject to} & \sum_i w_i \mathcal{I}_{\text{fair}}\ (z_i) \leq -(1-\beta)\ell^{\nu}_{\text{fair}}\ , \\ & \sum_i w_i \mathcal{I}_{\text{util}}\ (z_i) \leq \gamma\left(\min_{\mathbf{v}} \sum_i v_i \mathcal{I}_{\text{util}}\ (z_i)\right), \\ & w_i \in [0,1]. \end{array} \tag{6}$$

Fairness Metric. We define the fairness metric used in our paper, Demographic Parity.

Definition 2 (Demographic Parity Gap (DP) Dwork et al. (2012))**.** *Given a data distribution $\mathcal{D}$ over $\bar{\mathcal{X}} \times \{0,1\}$ from which features $x^{\setminus a}$ and sensitive attribute $x^a \in \{0,1\}$ are jointly drawn from, Demographic Parity gap for a model f_θ is defined to be the difference in the rate of positive predictions between the two groups, $|\Pr(\hat{y} \mid x^a = 0) - \Pr(\hat{y} \mid x^a = 1)|$ where $\hat{y}$ is the prediction $f_\theta(x^{\setminus a}, x^a)$.*

Dataset	Predict	Train / Val. / Test Split	#Dim.	Sensitive Attribute	Group Pos. Rate
Adult	if annual income $\geq$ 50k	22622/7540/15060	102	Gender (Male / Female)	0.312/0.113
Compas	if defendant rearrested in 2 yrs.	3700/1234/1233	433	Race (White / Non-white)	0.609/0.518
German Credit	good/bad credit risk	600/200/200	56	Age (30)	0.742/0.643

Table 3: Details for datasets used in our Fairness Manipulation attack experiments. In our setting, the Validation set is the test set shared between model training and influence calculation. The Test set is an untouched set which is used to assess the performance and fairness of the final model achieved after training with the reweighed training set. All our results are reported on this untouched Test set. Table details borrowed from Li and Liu (2022).

Dataset	Model ℓ_2 reg.	DP (β, γ)	EOP (β, γ)
Adult	2.26	(0.8,0.3)	(0.5,0.2)
Compas	37.00	(0.3,0.1)	(0.3,0.1)
German Credit	5.85	(0.5,0.0)	(0.5,0.0)

Table 4: Details for the Fairness Manipulation attack experiments. We use the same values of DP parameters as used by Li and Liu (2022).

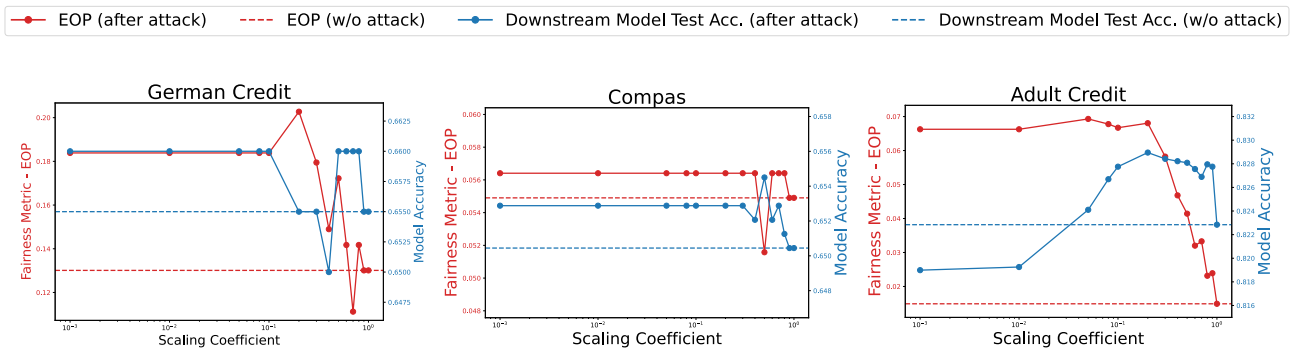

Figure 7: Scaling attack for the Fairness use-case. *Equal Opportunity Gap of post-attack downstream models is higher than that of those w/o attack while test accuracies are comparable. This implies that post-attack downstream models are less fair than those w/o attacks.* Scaling coefficients in log scale.