# Exploring the Limits of Model-Targeted Indiscriminate Data Poisoning Attacks

**Yiwei Lu** [1 2] **Gautam Kamath** [1 2] **Yaoliang Yu** [1 2]

## Abstract

Indiscriminate data poisoning attacks aim to decrease a model's test accuracy by injecting a small amount of corrupted training data. Despite significant interest, existing attacks remain relatively ineffective against modern machine learning (ML) architectures. In this work, we introduce the notion of *model poisoning reachability* as a technical tool to explore the intrinsic limits of data poisoning attacks towards target parameters (i.e., model-targeted attacks). We derive an easily computable threshold to establish and quantify a surprising phase transition phenomenon among popular ML models: data poisoning attacks can achieve certain target parameters only when the poisoning ratio exceeds our threshold. Building on existing parameter corruption attacks and refining the Gradient Canceling attack, we perform extensive experiments to confirm our theoretical findings, test the predictability of our transition threshold, and significantly improve existing indiscriminate data poisoning baselines over a range of datasets and models. Our work highlights the critical role played by the poisoning ratio, and sheds new insights on existing empirical results, attacks and mitigation strategies in data poisoning. Our code is available at `https://github.com/watml/plim`.

## 1. Introduction

Modern machine learning (ML) models require a large amount of training data to perform well on various tasks. Such hunger for data not only increases the training cost but also introduces potential risks during the data collection process (Kumar et al. 2020; Nelson et al. 2008; Szegedy et al. 2014). Data poisoning, where an adversary can actively inject corrupted data into dataset aggregators or passively place poisoned samples on the web for scraping (Gao et al. 2020; Lyu et al. 2020; Shejwalkar et al. 2022; Wakefield 2016), has caused serious concerns in the ML community and inspired a number of interesting works to expose and address this threat (Goldblum et al. 2023).

By now many data poisoning algorithms have been proposed; see Section 2 for some pointers. However, in the setting of indiscriminate data poisoning, where an attacker aims to decrease the overall test accuracy by adding a small fraction of corrupted data, the effectiveness of existing attacks remains underwhelming. For example, the recent work of Lu et al. (2022) achieved 1.11% accuracy drop for a three-layer CNN on MNIST and a 5.54% accuracy drop for ResNet-18 on CIFAR-10, after adding $\varepsilon_d = 3\%$ poisoned data and retraining. Part of the difficulty lies in the computational challenge: the attacker has to anticipate what would happen after retraining the model on the mixed data (clean in-house data plus poisoned data). Other empirical works seem to suggest there might also be some intrinsic barrier to data poisoning; see Section 2 for a detailed discussion.

In this work we focus on model-targeted attacks (e.g., Koh et al. 2022; Suya et al. 2021) and introduce the notion of *model poisoning reachability*, i.e., given (arbitrary) clean training data, what model, represented by its parameter $\mathbf{w}$, can be achieved through data poisoning, and what is the minimum (relative) percentage $\varepsilon_d$ of poisoned data that one has to introduce, with what algorithm? While model poisoning reachability intuitively depends on the clean training data, the loss and the target model we aim to achieve, we show that under mild conditions, it can be largely characterized by a simple threshold $\tau$ that is readily computable and involves no training at all. In particular, when the poisoning percentage $\varepsilon_d$ falls under $\tau$, no algorithm could achieve the target model by retraining on a mixed dataset (however crafted). On the flip side, if $\varepsilon_d > \tau$, we show that Gradient Canceling (GC), a refinement of the KKT attack of Koh et al. (2022), can achieve a given target model surprisingly efficiently. We further demonstrate that most ML classifiers exhibit a phase transition: they become poisoning reachable only when $\varepsilon_d$ crosses the threshold $\tau$. In contrast, regression methods can be poisoning reachable even when $\varepsilon_d$ approaches 0. Thus, our results expose the critical role played by the poisoning percentage $\varepsilon_d$, and clarify the somewhat disparate empirical results in the literature (with varying $\varepsilon_d$).

Authors GK and YY are listed in alphabetical order. [1]School of Computer Science, University of Waterloo, Canada [2]Vector Institute. Correspondence to: Yiwei Lu <yiwei.lu@uwaterloo.ca>.

*Proceedings of the $40^{th}$ International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. 

Empirically, we apply the GC attack and verify the *model poisoning reachability* property across a wide range of ML models, from logistic regression to residual networks on various datasets. Moreover, our work can also be applied as a distillation device: given any target parameter (namely the teacher, however crafted or impractical) that is effective for certain purpose, we can use our threshold and GC attack to pinpoint the (minimum) amount of poisoning data that needs to be constructed in order to simulate the teacher through retraining the model (student) over the combination of clean and poisoned data. Indeed, using the target parameters generated by parameter corruption (Sun et al. 2020) as a teacher, GC is able to construct more practical and effective (student) data poisoning attacks than baseline methods.

We summarize our main contributions as follows:

- We formalize the notion of model poisoning reachability as a technical tool to study model-targeted data poisoning and we derive an easily computable threshold to characterize it.
- We quantify the critical role played by the poisoning ratio $\varepsilon_d$ and we establish a surprising phase transition for ML classifiers, explaining seemingly disparate empirical results obtained with varying $\varepsilon_d$.
- We perform the Gradient Canceling attack on a number of models and datasets to extensively test our results. With carefully chosen target parameters, we are able to improve existing indiscriminate data poisoning baselines.

## 2. Background

Data poisoning, an emerging concern on modern ML systems, refers to the threat of (often passively) crafting "poisoned" training data so that systems retrained on it (along with possibly clean in-house data) are skewed towards certain behaviour. For example, indiscriminate data poisoning (e.g., Biggio et al. 2012; Koh and Liang 2017; Koh et al. 2022; Lu et al. 2022; Muñoz-González et al. 2017) aims to decrease the overall test accuracy while targeted data poisoning (e.g., Aghakhani et al. 2021; Guo and Liu 2020; Shafahi et al. 2018; Zhu et al. 2019) only affects certain classes. Backdoor attacks (e.g., Chen et al. 2017; Gu et al. 2017; Saha et al. 2020; Tran et al. 2018) that aim to trigger a particular pattern, and unlearnable examples (e.g., Fowl et al. 2021a,b; Fu et al. 2021; Huang et al. 2021; Liu and Chawla 2010; Sandoval-Segura et al. 2022; Yu et al. 2022) that aim to protect user data.

While many algorithms have been proposed for data poisoning, their effectiveness remains largely underwhelming against neural networks, especially when $\varepsilon_d$, the relative proportion of poisoned data, is small. For example, Figure 4 of Lu et al. (2022) and Table 2 of Huang et al. (2021) revealed that SOTA attacks can only decrease the test accuracy noticeably when $\varepsilon_d$ is sufficiently (and sometimes exceedingly, e.g., $\varepsilon_d > 100\%$) large. These attacks, relying on sophisticated optimization tricks, are also rather expensive to run. On the other hand, any data poisoning attack amounts to an *indirect* way of rewiring an ML model (i.e., any change must be induced by retraining the model over clean and poisoned data). Direct approaches, such as the gradient-based parameter corruption (GradPC) attack of Sun et al. (2020) and Zhang et al. (2021), seek to overwrite a target model *directly* (i.e., without constructing any poisoned data or retraining), under a perturbation constraint specified by $\varepsilon_w$, i.e., the relative change of the model parameter should be less than $\varepsilon_w$. While the applicability of direct approaches may seem limited, they are suitable for exploring the limits of more realistic data poisoning attacks.

Table 1: The attack accuracy/accuracy drop (%) on MNIST.

| Model | Clean | TGDA | GradPC | |
|---|---|---|---|---|
| | Acc. | Accuracy/Drop | $\varepsilon_w = 0.5$ | $\varepsilon_w = 1$ |
| LR | 92.35 | 89.56 / 2.79 ($\varepsilon_w = 2.45$) | 69.80 / 22.55 | **21.48 / 70.87** |
| NN | 98.04 | 96.54 / 1.50 ($\varepsilon_w = 0.55$) | 76.51 / 20.03 | **31.14 / 66.90** |
| CNN | 99.13 | 98.02 / 1.11 ($\varepsilon_w = 0.74$) | 73.24 / 24.78 | **12.98 / 86.15** |

In Table 1 we compare the performance of the direct approach GradPC (Sun et al. 2020) and the indirect approach TGDA (Lu et al. 2022). The latter adds $\varepsilon_d = 3\%$ poisoned data while both attacks yield comparable perturbations of the (clean) model, as measured by $\varepsilon_w$. The difference is significant, and begs the obvious question: what caused this difference? Is it because existing data poisoning attacks are not sufficiently optimized yet, or is there some intrinsic barrier to produce certain target parameters through data poisoning? To what extent would increasing $\varepsilon_d$ help, and how do we know without trying every $\varepsilon_d$? These questions will be formally and experimentally explored in the sequel, with the ultimate goal (if possible) to reduce the gap between data poisoning and parameter corruption attacks with comparable $\varepsilon_w$, as highlighted in Table 1.

**Connection with Learning Theory:** There has been significant work on training-time robustness in the learning theory literature, primarily focused on poisoning *worst-case distributions*. Two models of robust PAC learning (Frénay and Verleysen 2014; Natarajan et al. 2013), slightly rephrased for the sake of comparison, include the malicious noise model, where the adversary adds points (e.g., Cesa-Bianchi et al. 1999; Kearns and Li 1988), and the nasty noise model, where the adversary may both add and remove points (e.g., Balcan et al. 2022; Bshouty et al. 2002). Many of these theoretical results show strong computational barriers against robust learning for even the most basic problems. Although our setting is similar to the malicious noise model (and we touch a bit on the nasty noise model in Appendix C.10), there are three major differences with the majority of the theory literature: (1) our attacks address distributions that arise in practice, which differ from worst-case distributions;

(2) while other attacks flip labels, we consider "clean label" attacks which are not visibly mislabeled; (3) we focus on model-targeted attacks whose goal is to induce certain target parameters while the above-mentioned references focus directly on decreasing accuracy on the test sample.

## 3. Theoretical Results

In this section we formalize the notion of model poisoning reachability as a technical tool for studying model-targeted data poisoning. We further derive an easily computable threshold $\tau$ and reveal that model-targeted data poisoning attacks are effective only when $\varepsilon_d$, the (relative) percentage of poisoning data, crosses $\tau$.

**Notation and Preliminaries.** Let $\ell(\mathbf{z}, \mathbf{w})$ be our loss that measures the fitness of our model $\mathbf{w}$ on data $\mathbf{z} \in \mathbb{Z}$, e.g., $\mathbf{z} = (\mathbf{x}, y)$ for supervised learning and $\mathbf{z} = \mathbf{x}$ for unsupervised learning. Let $\mathcal{P} = \mathcal{P}(\mathbb{Z})$ denote the set of all distributions on $\mathbb{Z}$, and we abstract the training data as an (empirical) distribution[1] $\mu \in \mathcal{P}$. For any given model $\mathbf{w}$ and training distribution $\mu$, is it possible to construct a poisoning set, denoted by another (empirical) distribution $\nu$, such that $\mathbf{w}$ minimizes $\ell$ over the mixed distribution $\chi = (1-\lambda)\mu + \lambda\nu$, where $\lambda = \frac{\varepsilon_d}{1+\varepsilon_d} \in [0, 1]$ is the proportion of poisoning data. To account for possible nonconvexity of the loss $\ell$, we relax the optimality of a model $\mathbf{w}$ to simply have vanishing (sub)gradient. More formally, let

$$\mathbf{g}(\mathbf{z}) = \mathbf{g}(\mathbf{z}; \mathbf{w}) = \nabla_{\mathbf{w}} \ell(\mathbf{z}; \mathbf{w}) \tag{1}$$

be the gradient vector with respect to a fixed model $\mathbf{w}$ evaluated at the data $\mathbf{z}$. For practical reasons (e.g., to evade possible defenses or to account for the technical capabilities of an attacker) we also restrict the poisoning distribution $\nu$ into a convex subset $\Gamma \subseteq \mathcal{P}$ of admissible distributions. For instance, we may consider

$$\Gamma = \Gamma_{\mu,\delta} := \{\gamma : \|\gamma - \mu\| \leq \delta\}, \tag{2}$$

where $\|\cdot\|$ denotes (say) the Wasserstein distance. By definition we always have $\mu \in \Gamma$. For each $\nu \in \Gamma$, define

$$\mathbf{g}(\nu) = \mathbf{g}(\nu; \mathbf{w}) := \mathbb{E}_{\mathbf{z}\sim\nu} \mathbf{g}(\mathbf{z}; \mathbf{w}), \tag{3}$$

i.e., the average gradient w.r.t. the distribution $\nu$. Clearly,

$$\mathbb{G} = \mathbb{G}(\Gamma) := \{\mathbf{g}(\nu) : \nu \in \Gamma\} \tag{4}$$

is a subset of the closed convex hull of all gradient vectors. In fact, equality holds when $\Gamma = \mathcal{P}$ (e.g., $\delta = \infty$).

### 3.1. Model Poisoning Reachability

We can now state our fundamental problem of interest:

**Definition 1** (Model Poisoning Reachability)**.** *We say a target parameter* $\mathbf{w}$ *is* $(\ell, \mu, \Gamma, \lambda)$*-poisoning reachable if there exists some poisoning distribution* $\nu \in \Gamma$ *such that*

$$\mathbf{g}(\chi; \mathbf{w}) = (1-\lambda)\mathbf{g}(\mu; \mathbf{w}) + \lambda \mathbf{g}(\nu; \mathbf{w}) = \mathbf{0}, \tag{5}$$

*i.e., the parameter* $\mathbf{w}$ *has vanishing gradient (w.r.t. loss* $\ell$*) over the mixed distribution* $\chi = (1-\lambda)\mu + \lambda\nu$*.*

When the loss $\ell$, training distribution $\mu$, and admissible poisoning distributions $\Gamma$ are evident, we will simply say the parameter $\mathbf{w}$ is $\lambda$-poisoning reachable, or poisoning reachable if it is $\lambda$-poisoning reachable for some $\lambda \in [0, 1]$.

We make three further remarks regarding Definition 1: (a) If we are interested in more quantitative results about data poisoning, for example, is it possible to craft a poisoning set such that retraining on the mixed distribution would decrease test accuracy by a large margin, we need only specify a set of target models $\mathbf{w} \in \mathcal{W}$ that all decrease the test accuracy as required[2], and we say data poisoning is successful if any $\mathbf{w} \in \mathcal{W}$ is ($\lambda$-) poisoning reachable. (b) Definition 1 leaves out the computational aspects of data poisoning, i.e., how efficiently we can find such a poisoning distribution $\nu$ (whenever it exists). This will be studied in Section 4, using a gradient-based algorithm inspired directly by our definition. (c) We could also add other requirements, such as curvature or stability, to Definition 1.

Given the above formalization, the following characterization is immediate:

**Theorem 1.** *A target parameter* $\mathbf{w}$ *is* $\lambda$*-poisoning reachable iff* $\mathbf{0} \in \mathbb{G}^\lambda = \mathbb{G}^\lambda(\mathbf{g}(\mu)) := \{(1-\lambda)\mathbf{g}(\mu) + \lambda\mathbf{g} : \mathbf{g} \in \mathbb{G}\}$.

Since $\mathbb{G}$ (see equations (1)-(4)) is clearly convex, the subsets $\mathbb{G}^\lambda$ are all convex and increasing with respect to $\lambda$, i.e.,

$$\mathbf{g}(\mu) = \mathbb{G}^0 \subseteq \mathbb{G}^\lambda \uparrow\subseteq \mathbb{G}^1 = \mathbb{G}.$$

Recall that $\lambda = \frac{\varepsilon_d}{1+\varepsilon_d}$ is the (absolute) proportion of the poisoned set. Thus, we conclude intuitively that the larger $\varepsilon_d$ (equivalently $\lambda$) is, the easier it is to induce any target model $\mathbf{w}$ on any training distribution $\mu$. In particular, the special case $\lambda = 1$ corresponds to the so-called "unlearnable examples" (Fowl et al. 2021a,b; Fu et al. 2021; Huang et al. 2021; Liu and Chawla 2010; Sandoval-Segura et al. 2022; Yu et al. 2022), where an attacker is allowed to change the entire training set (i.e., empirical distribution $\mu$).

Conversely, we can also conclude from Theorem 1 that if $\mathbf{0} \notin \mathbb{G}(\Gamma)$, then data poisoning, with any budget $\varepsilon_d$, will not be successful in producing the target parameter $\mathbf{w}$. If $\mathbf{0} \notin \mathbb{G}(\mathcal{P})$, then no training distribution can yield $\mathbf{w}$. In particular, data poisoning will not be successful in producing $\mathbf{w}$ even if $\varepsilon_w = \infty$.

[1] For convenience in this work we do not distinguish the (clean) training set from the training distribution, i.e., $\mu$ can be empirical.

[2] As pointed out by a reviewer, this may not be computationally feasible if one is too ambitious about the set $\mathcal{W}$.

Let us give some examples to illustrate our results so far.

**Example 1** (Least-square regression)**.** *Consider*

$$\ell(\mathbf{z};\mathbf{w}) = \tfrac{1}{2}(y - \mathbf{w}^\top \mathbf{x})^2, \text{ where } \mathbf{z} = (\mathbf{x}, y).$$

*Clearly, we have*

$$\mathbf{g}(\mathbf{x}, y) = (\mathbf{w}^\top \mathbf{x} - y)\mathbf{x} = (\mathbf{x}\mathbf{x}^\top)\mathbf{w} - y\mathbf{x},$$

*and hence* $\mathbf{g}(\mu) = \Sigma\mathbf{w} - \mathbf{m}$*, where* $\Sigma = \mathbb{E}_{\mathbf{x}\sim\mu}\mathbf{x}\mathbf{x}^\top$ *and* $\mathbf{m} = \mathbb{E}_{(\mathbf{x},y)\sim\mu} y\mathbf{x}$*. For simplicity let us assume* $\mathbb{Z} = \mathbb{R}^d \times \mathbb{R}$ *and* $\Gamma = \mathcal{P}$ *so that* $\mathbb{G} = \mathbb{R}^d$ *(by considering product distributions where* $\mathbf{x}$ *concentrates on a single point). Therefore, we conclude from Theorem 1 that data poisoning with any* $\varepsilon_d > 0$ *is possible for least-square regression. The same conclusion holds even if we add regularization to* $\mathbf{w}$ *(which, we recall, is fixed).*

### 3.2. Scalar Output Linear Models

For linear models we can further simplify the iff condition in Theorem 1. We begin with the following result:

**Theorem 2.** *Suppose* $\Gamma = \mathcal{P}$ *contains all distributions,* $\ell((\mathbf{x}, y); \mathbf{w}) = l(\mathbf{w}^\top \mathbf{x}, y)$ *for some univariate loss* $l$*, and* $\langle \mathbf{w}, \mathbf{g}(\mu)\rangle \neq 0$*. Then,* $\mathbf{w}$ *is* $\lambda$*-poisoning reachable iff*

$$\lambda > \max\left\{\tfrac{\langle \mathbf{w},\mathbf{g}(\mu)\rangle}{\langle \mathbf{w},\mathbf{g}(\mu)\rangle - a}, \tfrac{-\langle \mathbf{w},\mathbf{g}(\mu)\rangle}{b - \langle \mathbf{w},\mathbf{g}(\mu)\rangle}\right\}, \quad \textit{where} \tag{6}$$

$$a = \inf_{(\mathbf{x},y)\in\mathbb{Z}} (\mathbf{w}^\top \mathbf{x}) \cdot l'(\mathbf{w}^\top \mathbf{x}, y),$$

$$b = \sup_{(\mathbf{x},y)\in\mathbb{Z}} (\mathbf{w}^\top \mathbf{x}) \cdot l'(\mathbf{w}^\top \mathbf{x}, y),$$

*with equality attained if the maximum is attained.*

Theorem 2 follows from the more general Theorem 5 in Appendix A, where we further remove the restriction $\Gamma = \mathcal{P}$. The condition $\langle \mathbf{w}, \mathbf{g}(\mu)\rangle \neq 0$ can be checked easily *a priori*; see Remark 1 (Appendix A) for discussions on when it fails. Remark 2 (Appendix A) draws further connection between our result and the breakdown point in robust statistics. Compared to the more general Theorem 1, Theorem 2 exploits the linear structure to simplify the set $\mathbb{G}$ to basically an interval and hence the condition (6) is much easier to verify. Indeed, consider Example 1 again. It is clear that $l'(t, y) = t - y$, whence $a = -\infty$ and $b = \infty$. Thus, we verify more easily that data poisoning succeeds on least-square regression for any $\varepsilon_d > 0$.

The next example reveals a surprising phase transition in terms of the poisoning proportion $\lambda$ (or equivalently $\varepsilon_d$):

**Example 2** (Logistic regression)**.** *Consider now*

$$\ell(\mathbf{z};\mathbf{w}) = \log(1 + \exp(-\mathbf{w}^\top \tilde{\mathbf{x}})),$$

*where we have absorbed the binary label* $y$ *into* $\tilde{\mathbf{x}}$ *(e.g.,* $\tilde{\mathbf{x}} \leftarrow y\mathbf{x}$*). Clearly, we have* $\mathbf{g}(\tilde{\mathbf{x}}) = -\frac{1}{1+\exp(\mathbf{w}^\top \tilde{\mathbf{x}})}\tilde{\mathbf{x}}$*. On the direction* $\mathbf{w}$*, for any distribution* $\mu$ *we have*

$$-\mathscr{W}(\tfrac{1}{e}) = \inf_t \tfrac{-t}{1+\exp(t)} \le \langle \mathbf{w}, \mathbf{g}(\mu)\rangle \le \sup_t \tfrac{-t}{1+\exp(t)},$$

*where the left-hand side is Lambert's W function and the right-hand side is clearly* $\infty$*. Therefore, suppose* $\mathbb{X} = \mathbb{R}^d$ *and* $\Gamma = \mathcal{P}$*, we have*

$$\mathbb{G} = \{\mathbf{g} : \mathbf{w}^\top \mathbf{g} \ge -\mathscr{W}(1/e) \approx -0.28\},$$

*which is not the entire space! Consequently, if*

$$\lambda < \tfrac{\langle \mathbf{w},\mathbf{g}(\mu)\rangle}{\langle \mathbf{w},\mathbf{g}(\mu)\rangle + \mathscr{W}(1/e)} \iff \varepsilon_d < \tau := \max\{\tfrac{\langle \mathbf{w},\mathbf{g}(\mu)\rangle}{\mathscr{W}(1/e)}, 0\}, \tag{7}$$

*then any poisoning distribution* $\nu$ *(with any support) cannot produce* $\mathbf{w}$ *(along with training distribution* $\mu$*)!*

By simply changing $\tilde{\mathbf{x}} \leftarrow y\mathbf{x}$ and then dropping $y$ we immediately obtain from Theorem 2 sufficient and necessary conditions for the poisoning reachability of binary margin classifiers. In particular, we record the following result:

**Corollary 1** (Binary Margin Classifier)**.** *Consider linear models with loss* $\ell(\tilde{\mathbf{x}}; \mathbf{w}) = l(\mathbf{w}^\top \tilde{\mathbf{x}})$*.*

*Suppose* $\Gamma = \mathcal{P}$ *consist of all distributions on* $\tilde{\mathbb{X}}$ *and* $\langle \mathbf{w}, \mathbf{g}(\mu)\rangle \neq 0$*. Define*

$$a := \inf_{t\in\mathbf{w}^\top\tilde{\mathbb{X}}} t \cdot l'(t), \quad b := \sup_{t\in\mathbf{w}^\top\tilde{\mathbb{X}}} t \cdot l'(t).$$

*Then, a target parameter* $\mathbf{w}$ *is* $\lambda$*-poisoning reachable iff* (6) *holds (with equality attained if the maximum there is attained).*

The standard margin losses are decreasing, such as the logistic loss in Example 2, the exponential loss in Adaboost, and the hinge loss in SVM. When $\mathbb{X} = \mathbb{R}^d$ is unbounded, typically $b = \infty$ but $a > -\infty$, leading to a common phase transition phenomenon: data poisoning against these losses succeeds in producing a target parameter $\mathbf{w}$ iff $\lambda$ crosses the threshold in (6). In particular, any target parameter $\mathbf{w}$ such that $\langle \mathbf{w}, \mathbf{g}(\mu)\rangle < 0$ is always poisoning reachable for any $\lambda > 0$. Interestingly, Koh et al. (2022, Proposition 3) showed that if a model is poisoning reachable, then it (often) can be poisoned to by a distribution $\nu$ supported on two distinct points (which however does not imply diminishing $\varepsilon_d$ due to repetitions). Corollary 1 provides a definitive answer on when a model is poisoning reachable and hence complements the results of Koh et al. (2022).

We emphasize that with any further restrictions on the poisoning distribution (such that $\Gamma \subsetneq \mathcal{P}$), condition (6) remains to be necessary: data poisoning is apparently even harder in this case. For nonlinear models with a fixed feature map $\phi$ (such as kernel methods), our results extend immediately, after the obvious change-of-variable $\mathbf{x} \leftarrow \phi(\mathbf{x})$.

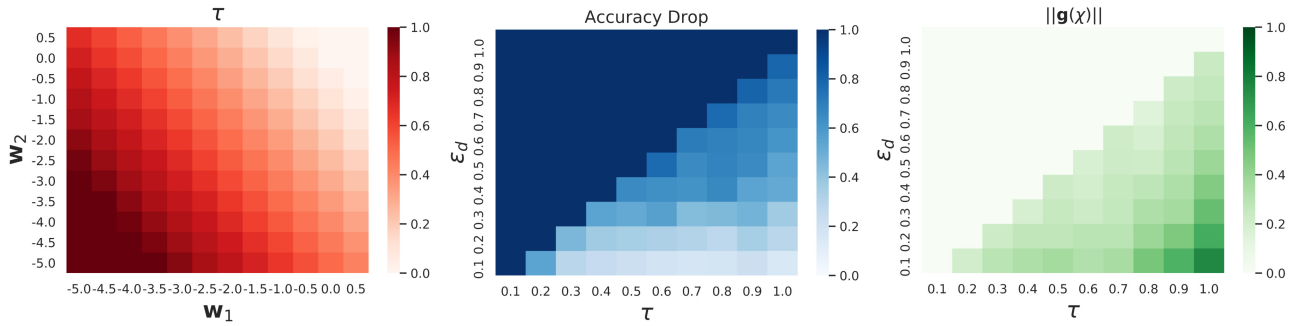


Figure 1: Logistic regression on the 2d OR dataset that verifies the transitioning threshold $\tau$ in Corollary 1. **Left**: $\tau$ w.r.t. target models $\mathbf{w} \in \mathbb{R}^2$, which all achieve 0 accuracy; **Middle**: accuracy drop due to the gradient canceling attack in Section 4. Indeed, poisoning successfully induces the target model $\mathbf{w}$ as long as $\varepsilon_d \geq \tau$; **Right**: norm of gradient w.r.t. model $\mathbf{w}$ over the mixed distribution $\chi$, with $\varepsilon_d$ the relative proportion of poisoned data. In general, the closer $\varepsilon_d$ gets above $\tau$, the smaller the gradient norm, which is an indication of the target model being more achievable through data poisoning.

Figure 1 illustrates the transition threshold $\tau$ in (7) on the simple OR dataset (where each of the four points is repeated 50 times with small Gaussian perturbation, see Appendix C for details). Logistic regression (LR), trained on the clean data, achieves perfect accuracy. In Figure 1 (left), each grid point represents a target parameter $\mathbf{w} = (w_1, w_2)$, all of which achieve 0 test accuracy (i.e., malicious models). The heat map indicates the threshold $\tau$ for each $\mathbf{w}$, which, as predicted by our theory, is the percentage of poisoning required to achieve $\mathbf{w}$ through retraining. In Figure 1 (middle) we run the gradient canceling attack (see Section 4) with varying percentage $\varepsilon_d$ and verify that indeed we can reduce the 100% clean accuracy to 0% iff $\varepsilon_d \geq \tau$. In Figure 1 (right) we plot the magnitude of the gradient of the target parameter $\mathbf{w}$ over the mixed dataset (clean training data plus poisoned data), as an approximate measure of how close $\mathbf{w}$ can be achieved by retraining on the mixed dataset. Overall, the larger $\varepsilon_d$ is, the larger the accuracy drop is (not surprisingly) and the smaller the gradient norm is, with a clear transition once $\varepsilon_d$ crosses $\tau$ (perhaps surprisingly).

### 3.3. Multiple Output Linear Models

Next, we extend our results to multiple outputs (classes):

**Theorem 3** (Multiclass)**.** *Consider* $\ell(\mathbf{x}, \mathbf{y}; W) = l(W^\top \mathbf{x}, \mathbf{y})$ *for some loss* $l$*. Then*[3]*,*

$$G(\mathbf{x}, \mathbf{y}) := \nabla_W \ell(\mathbf{x}, \mathbf{y}; W) = \mathbf{x} \otimes \nabla l(W^\top \mathbf{x}, \mathbf{y}). \quad (8)$$

*Suppose* $W^\top G(\mu)$ *is non-degenerate and* $\Gamma = \mathcal{P}$ *contains all distributions. Then,* $W$ *is* $\lambda$*-poisoning reachable iff*

$$\mathbf{0} \in (1-\lambda) W^\top G(\mu) + \lambda \{W^\top G(\nu) : \nu \in \Gamma\}. \quad (9)$$

Compared to Theorem 5, condition (9) is no longer univariate but a square matrix of dimensions the same as $\mathbf{y}$ (the output). Nevertheless, we may simply take the trace on both sides to arrive at an easier albeit only necessary condition. We illustrate the last point through a familiar example:

**Example 3** (Cross-entropy)**.** *Let* $\mathbf{h} = W^\top \mathbf{x}$*. The cross-entropy loss corresponds to*

$$l(\mathbf{h}, \mathbf{y}) = -\langle \mathbf{h}, \mathbf{y} \rangle + \log \sum\nolimits_k \exp h_k,$$

*where* $\mathbf{y}$ *is one-hot. Taking trace on* (9) *we obtain*

$$0 = (1-\lambda) g(\mu) + \lambda g(\nu), \text{ where}$$
$$g(\nu) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\nu} \langle \mathbf{h}, \mathbf{p} - \mathbf{y} \rangle,$$

*and* $\mathbf{p} := \mathrm{softmax}(\mathbf{h}) = \exp(\mathbf{h}) / \sum_k \exp(h_k)$*. In Appendix A we prove the tight bound* $-\mathscr{W}(\frac{c-1}{e}) \leq g(\nu) \leq \infty$*, leading to the necessary condition for inducing* $W$*:*

$$\varepsilon_d \geq \tau = \tau(c) := \max\{\langle W, G(\mu) \rangle / \mathscr{W}(\tfrac{c-1}{e}), 0\}, \quad (10)$$

*where* $c$ *is the number of classes. When* $c = 2$*, we recover the sufficient and necessary condition in* (7)*.*

We remark that all of our results continue to hold as necessary (but may not be sufficient) conditions for neural networks where the input $\mathbf{x}$ goes through a *learned* feature transformation $\varphi(\mathbf{x}; \mathbf{u})$, parameterized by $\mathbf{u}$:

**Theorem 4** (Neural Networks)**.** *Consider* $\ell(\mathbf{x}, \mathbf{y}; W, \mathbf{u}) = l(\mathbf{h}, \mathbf{y})$ *for some loss* $l$*, where* $\mathbf{h} := W^\top \varphi(\mathbf{x}; \mathbf{u})$*. Then,*

$$\nabla_W \ell(\mathbf{x}, \mathbf{y}; W, \mathbf{u}) = \varphi(\mathbf{x}; \mathbf{u}) \otimes \nabla_{\mathbf{h}} l(\mathbf{h}, \mathbf{y}) \quad (11)$$
$$\nabla_{\mathbf{u}} \ell(\mathbf{x}, \mathbf{y}; W, \mathbf{u}) = \nabla_{\mathbf{u}} \varphi(\mathbf{x}; \mathbf{u}) W \nabla_{\mathbf{h}} l(\mathbf{h}, \mathbf{y}), \quad (12)$$

*and* $(W, \mathbf{u})$ *is* $\lambda$*-poisoning reachable iff there exists* $\nu \in \Gamma$ *such that*

$$\mathbf{0} \in (1-\lambda) G(\mu) + \lambda G(\nu), \quad (13)$$

*where* $G(\nu) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\nu} (\nabla_W \ell, \nabla_{\mathbf{u}} \ell)$*. In particular,* $(W, \mathbf{u})$ *is* $\lambda$*-poisoning reachable only if there exists some* $\nu \in \Gamma$ *such that*

$$\mathbf{0} \in (1-\lambda) G_1(\mu) + \lambda G_1(\nu), \quad (14)$$

*where* $G_1(\nu) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\nu} \varphi(\mathbf{x}; \mathbf{u}) \otimes \nabla_{\mathbf{h}} l(\mathbf{h}, \mathbf{y})$.

[3] We use the notation $\mathbf{a} \otimes \mathbf{b} := \mathbf{a}\mathbf{b}^\top$ for two column vectors.

## 4. Gradient Canceling Attack

In this section we discuss how to find a poisoning distribution $\nu \in \Gamma$ so that upon retraining on the mixed distribution $\chi = (1-\lambda)\mu + \lambda\nu$, the target parameter $\mathbf{w}$ will be favored. We recall that $\mu$ is the (clean) training distribution and $\lambda$ is the (absolute) poisoning proportion.

The algorithm we propose is very intuitive and directly inspired by our Definition 1: we simply find a poisoning distribution $\nu \in \Gamma$ so that

$$\mathbf{g}(\chi) = (1-\lambda)\mathbf{g}(\mu) + \lambda\mathbf{g}(\nu) \approx \mathbf{0}, \tag{15}$$

where recall that $\lambda = \frac{\varepsilon_d}{1+\varepsilon_d}$ and $\mathbf{g}(\nu) := \mathbb{E}_{\mathbf{z}\sim\nu}\nabla_{\mathbf{w}}\ell(\mathbf{z};\mathbf{w})$ is the model gradient computed over a distribution $\nu$. Thus, we arrive at the following Gradient Canceling problem[4]:

$$\min_{\nu\in\Gamma} \ \tfrac{1}{2}\|\mathbf{g}(\mu) + \varepsilon_d\mathbf{g}(\nu)\|_2^2, \tag{16}$$

which is always convex (since $\mathbf{g}(\nu)$ is linear in $\nu$ while $\Gamma$ is a convex subset of admissible distributions). In Appendix B we discuss a measure optimization approach for solving (16), while below we focus on a Lagrangian approach that directly constructs a poisoning dataset and eliminates the need of resampling from $\nu$.

In more details, we constrain the poisoning distribution to be uniform over $n\varepsilon_d$ data points $\{\mathbf{z}_j\}$:

$$\hat{\nu} = \frac{1}{n\varepsilon_d}\sum_{j=1}^{n\varepsilon_d}\delta_{\mathbf{z}_j}, \tag{17}$$

where $n$ is the size of the (clean) training set and $\delta_{\mathbf{z}}$ denotes the point mass concentrated on $\mathbf{z}$. We only optimize the locations of the points $\mathbf{z}_j$ but keep their mass uniform throughout.

Thus, we arrive at the following problem:

$$\min_{\hat{\nu}\in\Gamma} \ \frac{1}{2}\Big\|\mathbf{g}(\mu) + \varepsilon_d \cdot \frac{1}{n\varepsilon_d}\sum_{j=1}^{n\varepsilon_d}\nabla_{\mathbf{w}}\ell(\mathbf{z}_j;\mathbf{w})\Big\|_2^2, \tag{18}$$

where we remind that $\mathbf{g}(\mu) = \mathbb{E}_{\mathbf{z}\sim\mu}\nabla_{\mathbf{w}}\ell(\mathbf{z};\mathbf{w})$ as well as the target parameter $\mathbf{w}$ are fixed during optimization. For supervised tasks where $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, we may choose to optimize both the feature $\mathbf{x}$ and label $\mathbf{y}$, or simply optimize the feature $\mathbf{x}$ only (as in our experiments).

We apply (projected) gradient descent to solve (18), where the gradient with respect to the $j$-th poisoning data $\mathbf{z}_j$ is:

$$\frac{\partial}{\partial\mathbf{z}_j} = \frac{1}{n}\nabla_{\mathbf{z}}\nabla_{\mathbf{w}}\ell(\mathbf{z}_j;\mathbf{w}) \cdot [\mathbf{g}(\mu) + \varepsilon_d\mathbf{g}(\hat{\nu})]. \tag{19}$$

We note that using auto-differentiation, the above matrix vector product can be computed very efficiently, costing essentially as much as gradient calculation. The constraint for $\hat{\nu}$ to lie in $\Gamma$ can be handled by projection. For instance, the constraint $\mathbf{z} \in \mathbb{Z}$ (e.g. pixels must lie in $\mathbb{Z} = [0,1]$) can be enforced by projecting the gradient update onto $\mathbb{Z}$.

[4]Other merit functions than the $\ell_2$-norm here can also be used.

**Algorithm 1:** Gradient Canceling(GC) Attack

**Input:** training distribution $\mu$, step size $\eta$, poisoning fraction $\varepsilon_d$, and target parameter $\mathbf{w}$.

initialize poisoned dataset $\hat{\nu}$ in (17), e.g., randomly subsample clean training data
calculate $\mathbf{g}(\mu) = \mathbb{E}_{\mathbf{z}\sim\mu}\nabla_{\mathbf{w}}\ell(\mathbf{z};\mathbf{w})$
**for** $t = 1, 2, \ldots$ **do**
  calculate $\mathbf{g}(\hat{\nu}) \leftarrow \frac{1}{n\varepsilon_d}\sum_{j=1}^{n\varepsilon_d}\nabla_{\mathbf{w}}\ell(\mathbf{z}_j;\mathbf{w})$
  calculate loss $\mathcal{L} = \frac{1}{2}\|\mathbf{g}(\mu) + \varepsilon_d\mathbf{g}(\hat{\nu})\|_2^2$
  update poisoned data using (19): $\mathbf{z}_j \leftarrow \mathbf{z}_j - \eta\frac{\partial\mathcal{L}}{\partial\mathbf{z}_j}$
  project to admissible set: $\hat{\nu} \leftarrow \mathrm{Proj}_{\Gamma}(\hat{\nu})$
**return** the final poisoned dataset $\hat{\nu}$

We summarize the Gradient Canceling(GC) attack in Algorithm 1, and we emphasize that it can take *any* target parameter $\mathbf{w}$ as "teacher" and construct a poisoning dataset such that retraining will arrive (approximately) at $\mathbf{w}$. We note that Gradient Canceling is a refinement of the KKT attack of Koh et al. (2022): our refinement lies in the generalization to any loss $\ell$, different optimization strategy, exploring target parameters generated by the much stronger GradPC attack (Sun et al. 2020), experimenting on a variety of different models, and studying the effect of the poisoning proportion. Other authors such as Suya et al. (2021) also explored (rather costly) attacks based on a target parameter in the online setting (that require retraining in each round), whereas their lower bound on the amount of poisoned points may not be easily computed even for logistic regression.

**Comparison with Gradient Matching.** Geiping et al. (2021) proposed a gradient matching algorithm for crafting *targeted* poisoning attacks, which can be easily adapted to our setting. Suppose that a defender aims at minimizing a loss $\ell$ to achieve model $\mathbf{w}$ on (clean) training distribution $\mu$. Let $\eth$ be a reversed version of $\ell$. For example, if $l$ is the cross-entropy loss in Example 3, then

$$l(\mathbf{h},\mathbf{y}) = -\log[1-\exp(-l(\mathbf{h},\mathbf{y}))], \text{ where } \mathbf{h} = W^\top\mathbf{x}, \tag{20}$$

is the reversed cross-entropy loss (Fowl et al. 2021a). As $\eth$ discourages the model from classifying clean data $\mathbf{x}$ as $\mathbf{y}$, Geiping et al. (2021) proposed to match its gradient $\nabla_{\mathbf{w}}\eth(\mu,\mathbf{w})$ over a poisoned distribution $\hat{\nu}$ (within some proximity of $\mu$), based on some dissimilarity function $\mathcal{S}$ (e.g., cosine dissimilarity):

$$\min_{\hat{\nu}\in\Gamma} \ \mathcal{S}\left(\nabla_{\mathbf{w}}\eth(\mu;\mathbf{w}),\ \nabla_{\mathbf{w}}\ell(\hat{\nu};\mathbf{w})\right). \tag{21}$$

We point out some key differences between gradient matching (Fowl et al. 2021a) and our work: (1) Gradient matching focuses on $\lambda = 1$, i.e., an attacker is able to modify the entire training set. While this is useful in certain settings (e.g., crafting "unlearnable examples"), it masks the effect of the poisoning proportion, which, as we showed in Section 3, can determine if a target parameter is poisoning reachable at all. (2) Gradient matching requires the construction of a reversed loss, whose gradient may not be at the same scale as that of the loss we are interested in. Thus, one typically can only hope to align the direction of gradients, which does not necessarily imply the desired matching in performance. In contrast, Algorithm 1 only requires the original loss and our theory gives guidance on when it succeeds. (3) There is no guarantee that after retraining over $\hat{\nu}$, gradient matching will arrive at the target parameter while Algorithm 1 explicitly aims to achieve this goal. Further experimental comparisons against gradient matching will be presented in Section 5 and Appendix C.

# 5. Experiments

We perform extensive experiments to verify our main results: (a) how competitive the gradient canceling attack (Algorithm 1) is compared to SOTA baselines in indiscriminate data poisoning? (b) to what extent our threshold $\tau$ (see (10)) can predict model poisoning reachability?(c) how effective gradient canceling remains against certain existing defense mechanisms?

## 5.1. Experimental Settings

**Dataset:** We consider image classification on MNIST (Deng 2012) (60k training and 10k test images), CIFAR-10 (Krizhevsky 2009) (50k training and 10k test images), and TinyImageNet (Chrabaszcz et al. 2017) (100k training, 10k validation and 10k test images). For the first two datasets, we further split the training data into 70% training set and 30% validation set, respectively.

**Target Models:** We examine the following ML models. On MNIST: Logistic Regression (LR), a fully connected neural network (NN) with three layers and a convolutional neural network (CNN) with two convolutional layers, max-pooling and one fully connected layer; On CIFAR-10: ResNet-18 (He et al. 2016); and on TinyImageNet: ResNet-34.

**Baselines:** We compare to TGDA (Lu et al. 2022) and Gradient Matching (Geiping et al. 2021) attacks. To our knowledge, the TGDA attack is one of the most effective data poisoning attacks against neural networks. Gradient Matching was originally proposed for targeted attacks and unlearnable examples, and we also compare against it due to its similarity with the Gradient Canceling (GC) attack.

**Implementation:** For GC implementation, we follow Algorithm 1 and we discuss the effect of the projection step in Section 5.4. Most of our target parameters are generated using GradPC [5] except LR on MNIST, where we use $\varepsilon_w = 1$ to allow meaningful accuracy drop and transition threshold $\tau$ [6]. We initialized the poisoned points with a random $n\varepsilon_d$ sample from the clean training set and we only optimized the feature vectors but not the labels. Accuracy drops are obtained after retraining.

**Evaluation Protocol:** To evaluate the effectiveness of different attacks, we first apply each attack to acquire its poisoned set and then retrain the model from scratch (initialized with the same random seed across all attacks) on both clean and poisoned data until convergence. The (test) accuracy drop, compared with clean accuracy (obtained by training on clean data only), is reported across all experiments.

## 5.2. How Competitive Is Gradient Canceling (GC)?

Table 2 reports the accuracy drop of LR, NN, CNN and ResNet due to GC on the aforementioned datasets. We note the trade-off of $\varepsilon_w$ in GradPC when generating a target parameter $\mathbf{w}$: the larger $\varepsilon_w$ is, the more effective GradPC is but also the larger the resulting transition threshold $\tau$ is, meaning that GC (or any other data poisoning attack) can succeed (in reproducing $\mathbf{w}$) only with a larger proportion $\varepsilon_d$ of poisoned points. We used $\tau = \tau(2)$ in Table 2 as we find it is much more indicative than the more conservative $\tau(c)$ (which is roughly 11 times smaller on TinyImageNet and 4 times smaller otherwise).

We observe that GC is much more effective than TGDA and Gradient Matching, across all datasets, models, and choices of $\varepsilon_d$. This confirms that existing data poisoning attacks are under-optimized and there is room for future improvements. Moreover, when $\varepsilon_d$ approaches the transition threshold $\tau$, GC, a *bona fide* data poisoning attack, indeed achieves a comparable accuracy drop as GradPC (which directly overwrites the model). While Table 1 still has room to improve, both in terms of the tightness of $\tau$ and the effectiveness of GC, we believe our results yield significant insights on indiscriminate data poisoning, in particular the theoretical and experimental quantification of the detrimental effect of a large proportion $\varepsilon_d$ of poisoned points.

## 5.3. Predicting Poisoning Reachability Using $\tau$

Next, we further examine the predictability of the transition threshold $\tau \approx \max\{3.6\,\langle W, G(\mu)\rangle, 0\}$, whose main term is simply proportional to the inner product between a target parameter and its gradient on the clean training data.

**Binary Logistic Regression**: We have already shown the

[5] We follow the implementation in `https://github.com/TobiasLee/ParamCorruption`.

[6] We discuss the selection of target parameters in Appendix D.

Table 2: The attack accuracy/accuracy drop (%) on MNIST, CIFAR-10 and TinyImageNet. We perform GC based on the target parameters generated by GradPC. Our attack significantly outperforms TGDA and Gradient Matching.

| Dataset | Target Model | Clean Acc | GradPC | Gradient Canceling | | | | TGDA | | | Gradient Matching | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\varepsilon_d$ | 0 | 0 | 0.03 | 0.1 | 1 | $\varepsilon_d = \tau$ | 0.03 | 0.1 | 1 | 0.03 | 0.1 | 1 |
| MNIST | LR | 92.35 | -70.87 ($\tau$=1.15) | **-22.97** | **-63.83** | **-67.01** | -69.66 | -2.79 | -4.01 | -8.97 | -3.33 | -8.14 | -12.13 |
| | NN | 98.04 | -20.03 ($\tau$=2.48) | **-6.10** | **-9.77** | **-12.05** | -19.05 | -1.50 | -1.72 | -5.49 | -2.82 | -3.71 | - 4.03 |
| | CNN | 99.13 | -24.78 ($\tau$=0.98) | **-9.55** | **-20.10** | **-23.80** | -23.77 | -1.11 | -1.31 | -4.76 | -2.01 | -3.80 | -6.94 |
| CIFAR-10 | ResNet-18 | 94.95 | -21.69 ($\tau$=1.29) | **-13.73** | **-16.40** | **-18.33** | -19.98 | -5.54 | -6.28 | -17.21 | -6.01 | -7.62 | -9.80 |
| TinyImageNet | ResNet-34 | 66.65 | -24.77 ($\tau$=1.08) | **-13.22** | **-16.11** | **-20.15** | -22.79 | -4.42 | -6.52 | -14.33 | -5.53 | -7.72 | -10.85 |

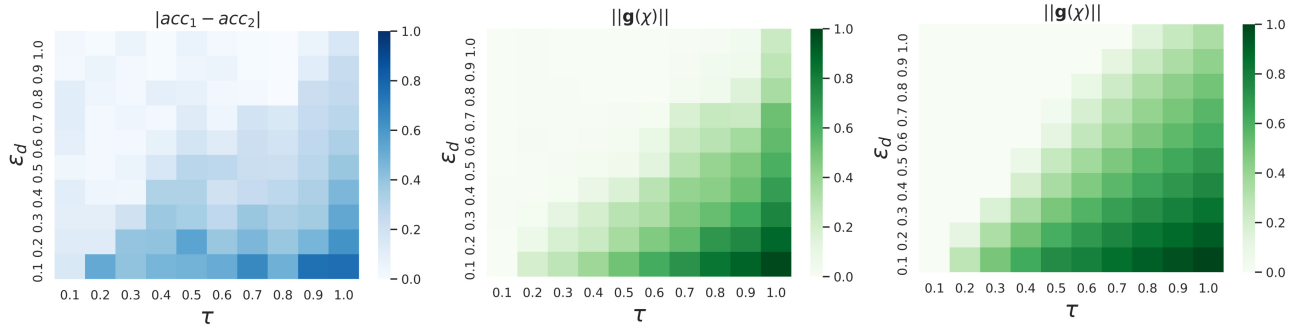


Figure 2: We run experiments on logistic regression to verify the transition threshold $\tau$ in Corollary 1. **Left**: accuracy difference between GC and GradPC on 10-d Gaussian dataset; **Middle**: norm of the gradient over the mixed dataset $\chi$ on 10-d Gaussian dataset; **Right**: norm of the gradient over the mixed dataset $\chi$ on MNIST-17.

predictability of $\tau$ on the OR dataset in Figure 1. In Figure 2 we show additional results on a 10-dimensional Gaussian dataset (see Appendix C.1) and MNIST-17 (consisting only of digits 1 and 7). The observations are similar: GC could achieve similar accuracy drops as GradPC (which directly overwrites the model), as long as $\varepsilon_d$ crosses the threshold $\tau$. We note that the threshold $\tau$ tends to be more conservative as the dimension of the problem increases, which we believe is largely because the optimization cost of GC becomes accordingly higher, making convergence harder to attain.

**Multi-class with Cross-Entropy:** We also perform experiments on multi-class problems with the cross-entropy loss in Example 3. In Table 2 we have confirmed that when $\varepsilon_d > \tau$, GC largely achieves the target parameters generated by GradPC. We now further examine the opposite case where $\varepsilon_d < \tau$. We fix $\varepsilon_d = 1$ and vary $\varepsilon_w$ in GradPC, consequently generating target parameters with varying $\tau$ on MNIST. Figure 3 shows how much the gradient $\|\mathbf{g}(\chi)\|$ of the target parameters decreases w.r.t. each epoch of GC (when $\chi$, the mixed dataset, gets updated). We observe that the gradients do not converge to 0, indicating that GC failed to produce the target parameters. The failure of GC indicates that a larger poisoning proportion $\varepsilon_d$ may be necessary to produce the target parameters, as confirmed by our theory.

### 5.4. Does GC Remain Effective Against Defenses?

Lastly, we choose several defenses from (Angel et al. 2022) and examine the effectiveness of GC against (1) a distribution-wise certified defense Sever (Diakonikolas et al. 2019), which removes $\varepsilon_d$ training points with the highest outlier scores, defined using the top singular value of the gradient matrix, and (2) one of the SOTA pointwise certified defenses (Levine and Feizi 2021; Wang et al. 2022a,b) called Deep Partition Aggregation (DPA) (Levine and Feizi 2021), which provides certified robustness for individual test samples. More results w.r.t. other defenses (e.g., influence defense and max-up defense) can be found in Appendix C.8.

Table 3: Accuracy drop (%) of Gradient Canceling (w/wo clipping) on MNIST against Sever defense (+ indicates the accuracy increased by the defense). GC-c: GC with clipped output; GC-d: GC after defense; GC-cd: GC-c after defense.

| **Model** | Clean | $\varepsilon_d$ | GC | GC-c | Sever | |
|---|---|---|---|---|---|---|
| | | | | | GC-d | GC-cd |
| LR | 92.35 | 0.03 | -22.79 | -11.28 | -12.81 / **+9.98** | -9.66 / **+1.62** |
| | | 0.1 | -63.83 | -26.77 | -59.79 / **+4.04** | -25.53 / **+1.24** |
| | | 1 | -67.01 | -28.99 | -65.01 / **+2.00** | -27.89 / **+1.10** |
| NN | 98.04 | 0.03 | -6.10 | -3.25 | -3.22 / **+2.88** | -2.26 / **+0.90** |
| | | 0.1 | -9.77 | -5.10 | -7.66 / **+2.11** | -4.46 / **+0.56** |
| | | 1 | -12.05 | -6.53 | -10.02 / **+2.03** | -6.11 / **+0.42** |
| CNN | 99.13 | 0.03 | -9.55 | -5.87 | -5.55 / **+4.00** | -4.36 / **+1.51** |
| | | 0.1 | -20.10 | -12.50 | -16.55 / **+3.55** | -11.32 / **+1.18** |
| | | 1 | -23.80 | -13.32 | -21.05 / **+2.75** | -12.51 / **+0.81** |

**Results on Sever:** Table 3 reports the accuracy drops on MNIST. We observe that (1) Sever indeed reduces the effectiveness of GC, consistently across all models. (2) Clipping poisoned data to the range of the clean training set makes GC more robust against all defenses, at the cost of less effectiveness in terms of accuracy drop. (3) Even with clipping and against defenses, GC still largely outperforms TGDA

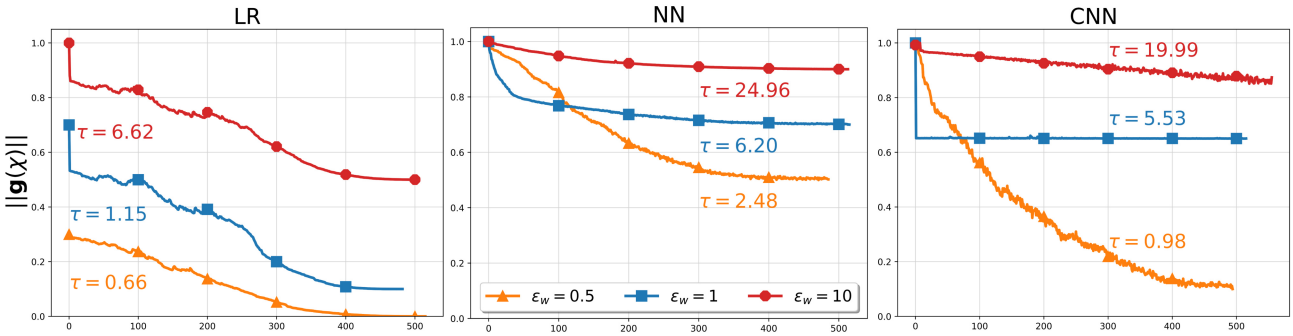


Figure 3: The learning curve for running GC on MNIST with different target models $\mathbf{w}$ and $\varepsilon_w$. We fix $\varepsilon_d = 1$, and the curves indicate the decrease of the gradient $\|\mathbf{g}(\chi)\|$ w.r.t. GC epoch. We confirm that GC fails to achieve $\mathbf{w}$ when $\varepsilon_d < \tau$.

Table 4: The Certified Accuracy (CA) (%) of DPA and Accuracy drop (%) of Gradient Canceling ($\varepsilon_d = 0.008$) on MNIST against the DPA defense (+ indicates the accuracy increased by the defense).

| **Model** | Clean | $k$ | Clean (DPA) | GC | CA | DPA |
|---|---|---|---|---|---|---|
| LR | 92.35 | 1200 | 91.33 | -8.25 | 47.12 | -4.68/**+3.57** |
| | | 3000 | 89.97 | -8.25 | 49.23 | -4.21/**+4.04** |
| NN | 98.04 | 1200 | 94.65 | -2.25 | 46.11 | -1.29/**+0.96** |
| | | 3000 | 92.37 | -2.25 | 48.52 | -1.17/**+1.08** |
| CNN | 99.13 | 1200 | 95.53 | -2.77 | 47.22 | -1.66/**+1.11** |
| | | 3000 | 93.15 | -2.77 | 50.01 | -1.52/**+1.25** |

and Gradient Matching. (4) Larger $\varepsilon_d$ generally makes GC both more effective and more robust, which matches our observation in least-squares regression (see Appendix C.3).

**Results on DPA**: although DPA is originally proposed for pointwise robustness, it can be easily applied to the indiscriminate data poisoning setting. Here we choose $k = 1200/3000$ for DPA and fix $\varepsilon_d = 0.008$ to roughly preserve median certified robustness on MNIST. Note that we choose the base classifiers to be the same as the target models. We report the certified accuracy (CA), which is the percentage of certified robust examples among the test set, and (relative) accuracy increase due to deploying DPA in Table 4. We observe that DPA is generally effective against GC, where the (relative) accuracy increased by DPA roughly approaches its certified accuracy. For example, on LR with $k = 3000$, GC was able to decrease test accuracy by $8.25\%$, whereas with the DPA defense, $4.04\%$ (relative) accuracy drop of GC are rectified, leading to an effectiveness that is roughly proportional to its certified accuracy, i.e., $4.04 \approx 8.25 * 0.4923$.

## 6. Conclusion and Future Work

In this work, we introduce the notion of *model poisoning reachability* as a technical tool to study the intrinsic limits in model-targeted data poisoning. We give complete characterizations on the poisoning ratio that any data poisoning attack has to satisfy (in order to induce a given target parameter), and we derive an easily computable threshold that is readily applicable and gives guidance on crafting effective model-targeted attacks. Using the gradient canceling attack, we perform extensive experiments on a number of datasets and models to quantify the critical role played by the poisoning ratio, confirm the precision of our transition threshold, and achieve better performance against existing baselines (w/wo several existing defenses). Our empirical results also reveal further room to sharpen the transition threshold and develop more effective data poisoning attacks, and we mention the exciting possibility of designing (clean) in-house data to mitigate and regulate the risk of future poisoning attacks.

One limitation of this work is its focus on achieving specific target parameters, which may not always be available or necessary. Indeed, data poisoning attacks that are not based on any target parameter abound. However, we point out that our work may still be valuable for the latter class of attacks, for instance, as a distillation device: a data poisoning attack can use our threshold to evaluate the potential "wastefulness" of its constructed poisoning set (along with the model parameter obtained by retraining) and then use GC to further distill and improve it. Another limitation is that most existing data poisoning attacks, including GC, assume a lot of knowledge of the victim model (e.g., fixed architecture, access to clean training data, etc.) and hence may not always be realistic. Advanced and adaptive defense mechanisms may also thwart the effectiveness of many attacks (including GC). Further investigations of these issues form another important direction for future research.

### Acknowledgments

We thank the reviewers for the critical comments that have largely improved the presentation and precision of this paper. We gratefully acknowledge funding support from NSERC and the Canada CIFAR AI Chairs program. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

## References


Aghakhani, H., D. Meng, Y.-X. Wang, C. Kruegel, and G. Vigna (2021). "Bullseye polytope: A scalable clean-label poisoning attack with improved transferability". In: *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 159–178.

Angel, N. B. et al. (2022). "Benchmarking the Effect of Poisoning Defenses on the Security and Bias of the Final Model". In: *NeurIPS Workshop on Trustworthy and Socially Responsible Machine Learning*.

Balcan, M.-F., A. Blum, S. Hanneke, and D. Sharma (2022). "Robustly-reliable learners under poisoning attacks". In: *Proceedings of Thirty Fifth Conference on Learning Theory*, pp. 4498–4534.

Biggio, B., B. Nelson, and P. Laskov (2012). "Poisoning attacks against support vector machines". In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pp. 1467–1474.

Bshouty, N. H., N. Eiron, and E. Kushilevitz (2002). "PAC learning with nasty noise". *Theoretical Computer Science*, vol. 288, no. 2, pp. 255–275.

Cesa-Bianchi, N., E. Dichterman, P. Fischer, E. Shamir, and H. U. Simon (1999). "Sample-efficient strategies for learning in the presence of noise". *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 684–719.

Chen, X., C. Liu, B. Li, K. Lu, and D. Song (2017). "Targeted backdoor attacks on deep learning systems using data poisoning". arXiv:1712.05526.

Chrabaszcz, P., I. Loshchilov, and F. Hutter (2017). "A downsampled variant of ImageNet as an alternative to the CIFAR datasets". arXiv preprint arXiv:1707.08819.

Deng, L. (2012). "The MNIST database of handwritten digit images for machine learning research". *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142.

Diakonikolas, I., G. Kamath, D. M. Kane, J. Li, J. Steinhardt, and A. Stewart (2019). "Sever: A Robust Meta-Algorithm for Stochastic Optimization". In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 1596–1606.

Fowl, L., P.-y. Chiang, M. Goldblum, J. Geiping, A. Bansal, W. Czaja, and T. Goldstein (2021a). "Preventing unauthorized use of proprietary data: Poisoning for secure dataset release". arXiv preprint arXiv:2103.02683.

Fowl, L., M. Goldblum, P.-y. Chiang, J. Geiping, W. Czaja, and T. Goldstein (2021b). "Adversarial Examples Make Strong Poisons". In: *Advances in Neural Information Processing Systems*, pp. 30339–30351.

Frénay, B. and M. Verleysen (2014). "Classification in the Presence of Label Noise: A Survey". *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869.

Fu, S., F. He, Y. Liu, L. Shen, and D. Tao (2021). "Robust unlearnable examples: Protecting data privacy against adversarial learning". In: *International Conference on Learning Representations*.

Gao, L. et al. (2020). "The Pile: An 800GB Dataset of Diverse Text for Language Modeling". arXiv preprint arXiv:2101.00027.

Geiping, J., L. H. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein (2021). "Witches' Brew: ial Scale Data Poisoning via Gradient Matching". In: *International Conference on Learning Representations*.

Goldblum, M., D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Madry, B. Li, and T. Goldstein (2023). "Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1563–1580.

Gu, T., B. Dolan-Gavitt, and S. Garg (2017). "Badnets: Identifying vulnerabilities in the machine learning model supply chain". arXiv:1708.06733.

Guo, J. and C. Liu (2020). "Practical Poisoning Attacks on Neural Networks". In: *European Conference on Computer Vision*, pp. 142–158.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Huang, H., X. Ma, S. M. Erfani, J. Bailey, and Y. Wang (2021). "Unlearnable Examples: Making Personal Data Unexploitable". In: *International Conference on Learning Representations*.

Kearns, M. and M. Li (1988). "Learning in the presence of malicious errors". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pp. 267–280.

Koh, P. W. and P. Liang (2017). "Understanding black-box predictions via influence functions". In: *Proceedings of*

*the 34th International Conference on Machine Learning (ICML)*, pp. 1885–1894.

Koh, P. W., J. Steinhardt, and P. Liang (2022). "Stronger Data Poisoning Attacks Break Data Sanitization Defenses". *Machine Learning*, vol. 111, pp. 1–47.

Krizhevsky, A. (2009). "Learning multiple layers of features from tiny images". tech. report.

Kumar, R. S. S., M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia (2020). "Adversarial machine learning-industry perspectives". In: *IEEE Security and Privacy Workshops (SPW)*, pp. 69–75.

Levine, A. and S. Feizi (2021). "Deep Partition Aggregation: Provable Defense against General Poisoning Attacks". In: *International Conference on Learning Representations*.

Liu, W. and S. Chawla (2010). "Mining adversarial patterns via regularized loss minimization". *Machine learning*, vol. 81, no. 1, pp. 69–83.

Lu, Y., G. Kamath, and Y. Yu (2022). "Indiscriminate Data Poisoning Attacks on Neural Networks". *Transactions on Machine Learning Research*.

Lyu, L., H. Yu, and Q. Yang (2020). "Threats to federated learning: A survey". arXiv preprint arXiv:2003.02133.

Muñoz-González, L., B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli (2017). "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec)*.

Natarajan, N., I. S. Dhillon, P. K. Ravikumar, and A. Tewari (2013). "Learning with noisy labels". *Advances in neural information processing systems*, vol. 26.

Nelson, B., M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia (2008). "Exploiting machine learning to subvert your spam filter." *LEET*, vol. 8, pp. 1–9.

Saha, A., A. Subramanya, and H. Pirsiavash (2020). "Hidden trigger backdoor attacks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sandoval-Segura, P., V. Singla, J. Geiping, M. Goldblum, T. Goldstein, and D. W. Jacobs (2022). "Autoregressive Perturbations for Data Poisoning". In: *Advances in Neural Information Processing Systems*.

Shafahi, A., W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein (2018). "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6103–6113.

Shejwalkar, V., A. Houmansadr, P. Kairouz, and D. Ramage (2022). "Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning". In: *IEEE Symposium on Security and Privacy (SP)*, pp. 1354–1371.

Sun, X., Z. Zhang, X. Ren, R. Luo, and L. Li (2020). "Exploring the vulnerability of deep neural networks: A study of parameter corruption". In: *Proceedings of the AAAI Conference on Artificial Intelligence*.

Suya, F., S. Mahloujifar, A. Suri, D. Evans, and Y. Tian (2021). "Model-targeted poisoning attacks with provable convergence". In: *Proceedings of the 38th International Conference on Machine Learning*, pp. 10000–10010.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2014). "Intriguing properties of neural networks". In: *International Conference on Learning Representation*.

Tran, B., J. Li, and A. Madry (2018). "Spectral Signatures in Backdoor Attacks". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Wakefield, J. (2016). "Microsoft chatbot is taught to swear on Twitter". *BBC News*.

Wang, W., A. Levine, and S. Feizi (2022a). "Lethal Dose Conjecture on Data Poisoning". In: *Advances in Neural Information Processing Systems*.

Wang, W., A. J. Levine, and S. Feizi (2022b). "Improved certified defenses against data poisoning with (deterministic) finite aggregation". In: *International Conference on Machine Learning*, pp. 22769–22783.

Yu, D., H. Zhang, W. Chen, J. Yin, and T.-Y. Liu (2022). "Availability Attacks Create Shortcuts". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2367–2376.

Zhang, Z., R. Luo, X. Ren, Q. Su, L. Li, and X. Sun (2021). "Adversarial parameter defense by multi-step risk minimization". *Neural Networks*, vol. 144, pp. 154–163.

Zhu, C., W. R. Huang, H. Li, G. Taylor, C. Studer, and T. Goldstein (2019). "Transferable clean-label poisoning attacks on deep neural nets". In: *International Conference on Machine Learning*, pp. 7614–7623.

## A. Proofs

**Theorem 5** (Linear Models)**.** *Consider $\ell((\mathbf{x}, y); \mathbf{w}) = l(\mathbf{w}^\top \mathbf{x}, y)$ for some univariate loss $l$. Then,*

$$\mathbf{g}(\mathbf{x}, y) = \mathbf{x} \cdot l'(\mathbf{w}^\top \mathbf{x}, y),$$

*and $\mathbf{w}$ is $\lambda$-poisoning reachable iff there exists $\nu \in \Gamma$ such that*

$$0 \in (1-\lambda)\mathbf{g}(\mu) + \lambda \mathbf{g}(\nu).$$

*Suppose $\langle \mathbf{w}, \mathbf{g}(\mu) \rangle \neq 0$. Consider $\Pi \subseteq \mathcal{P}$ and let*

$$\mathbb{J} := \{\mathbb{E}_{(\mathbf{x},y)\sim\nu}(\mathbf{w}^\top \mathbf{x}) \cdot l'(\mathbf{w}^\top \mathbf{x}, y) : \nu \in \Pi\} \subseteq \mathbb{R}.$$

*Then, $\mathbf{w}$ is $\lambda$-poisoning reachable if*[7] *$\Gamma \supseteq T_{\#}\Pi$ and*

$$0 \in (1-\lambda)\langle \mathbf{w}, \mathbf{g}(\mu) \rangle + \lambda \mathbb{J}, \tag{A.1}$$

*where the transformation $T(\mathbf{x}, y) := \left(\frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\langle \mathbf{w}, \mathbf{g}(\mu) \rangle} \mathbf{g}(\mu), y\right)$. Conversely,* (A.1) *holds if $\mathbf{w}$ is $\lambda$-poisoning reachable and $\Pi \supseteq \Gamma$.*

*Proof.* The gradient computation is straightforward while the first claim follows from Theorem 1.

Suppose now $\langle \mathbf{w}, \mathbf{g}(\mu) \rangle \neq \mathbf{0}$.

Suppose first (A.1) holds, so we can choose $\nu \in \Pi$ such that

$$0 = (1-\lambda)\langle \mathbf{w}, \mathbf{g}(\mu) \rangle + \lambda \mathbb{E}_{(\mathbf{x},y)\sim\nu}(\mathbf{w}^\top \mathbf{x}) \cdot l'(\mathbf{w}^\top \mathbf{x}, y). \tag{A.2}$$

Consider the transformation $T(\mathbf{x}, y) = \left(\frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\langle \mathbf{w}, \mathbf{g}(\mu) \rangle} \mathbf{g}(\mu), y\right)$ and let $\tilde{\nu} = T_{\#}\nu$, which is in $\Gamma$ due to our assumption $\Gamma \supseteq T_{\#}\Pi$. We then have

$$\mathbb{E}_{(\tilde{\mathbf{x}},\tilde{y})\sim\tilde{\nu}} l'(\mathbf{w}^\top \tilde{\mathbf{x}}, \tilde{y})\tilde{\mathbf{x}} = \mathbb{E}_{(\mathbf{x},y)\sim\nu} l'(\mathbf{w}^\top \mathbf{x}, y) \frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\langle \mathbf{w}, \mathbf{g}(\mu) \rangle} \mathbf{g}(\mu), \tag{A.3}$$

and hence

$$(1-\lambda)\mathbf{g}(\mu) + \lambda \mathbb{E}_{(\tilde{\mathbf{x}},\tilde{y})\sim\tilde{\nu}} \nabla \ell((\tilde{\mathbf{x}}, \tilde{y}); \mathbf{w}) = [(1-\lambda)\langle \mathbf{w}, \mathbf{g}(\mu) \rangle + \lambda \mathbb{E}_{(\mathbf{x},y)\sim\nu} l'(\mathbf{w}^\top \mathbf{x}, y) \langle \mathbf{w}, \mathbf{x} \rangle] \frac{\mathbf{g}(\mu)}{\langle \mathbf{w}, \mathbf{g}(\mu) \rangle} \tag{A.4}$$

$$= \mathbf{0}, \tag{A.5}$$

thanks to our choice of $\nu$. Applying Theorem 1 again we know $\mathbf{w}$ is $\lambda$-poisoning reachable.

Conversely, if $\mathbf{w}$ is $\lambda$-poisoning reachable, then from Theorem 1 it follows that

$$\mathbf{0} \in (1-\lambda)\mathbf{g}(\mu) + \lambda \mathbb{E}_{(\mathbf{x},y)\sim\nu} l'(\mathbf{w}^\top \mathbf{x}, y)\mathbf{x}. \tag{A.6}$$

Taking inner product with the model $\mathbf{w}$ on both sides and noting that $\Gamma \subseteq \Pi$ we verify (A.1). □

**Remark 1.** *The condition $\langle \mathbf{w}, \mathbf{g}(\mu) \rangle \neq 0$ can be easily checked* a priori*. In case it fails, two possibilities arise:*

- *$\mathbf{g}(\mu) = \mathbf{0}$, in which case poisoning is trivial: simply let $\nu = \mu$ for any $\lambda$.*
- *$\mathbf{g}(\mu) \neq \mathbf{0}$, in which case we may let $\nu$ concentrate on the line $L := \{\alpha \mathbf{g}(\mu) : \alpha \in \mathbb{R}\}$. Thus, data poisoning succeeds if*

$$0 = (1-\lambda) + \lambda \mathbb{E}_{(\alpha,y)\sim\nu} l'(0, y)\alpha, \tag{A.7}$$

*where we identify $\alpha \mathbf{g}(\mu)$ as $\alpha$ for $\nu$. As long as $\Gamma$ contains some distribution that puts nonzero mass on $L$ and sufficiently large $l'(0, y)$, $\mathbf{w}$ is again $\lambda$-poisoning reachable.*

[7] $T_{\#}\nu$ denotes the distribution of $T(\mathbf{z})$ when $\mathbf{z} \sim \nu$.

Once we identify an appropriate subset $\Pi$ of poisoning distributions, we can even estimate the interval $\mathbb{J}$ using Monte Carlo algorithms. Moreover, we may restrict the search of a poisoning distribution to the potentially much smaller subset $T_{\#}\Pi$ (where $\mathbf{x}$ lies on the line spanned by $\mathbf{g}(\mu)$).

**Remark 2** (Connection to breakdown point)**.** *For simplicity consider $\mathbb{Z} = \mathbb{R}^d \times \mathbb{R}$. It is well-known that unbounded convex losses $(t, y) \mapsto l(t - y)$, such as the square loss in Example 1, have 0 breakdown point (and hence not robust): even adding a single poisoning point can perturb the model norm $\|\mathbf{w}\|$ unboundedly (e.g. Yu et al. 2012, Theorem 5). Theorem 2 gives a much more detailed characterization: In fact, any target model $\mathbf{w}$ can be induced by a diminishing amount of poisoning (even if $\nu$ is supported on a single point)! Indeed, since $l$ is unbounded and convex, there exists some $\tau \in \mathbb{R}$ such that $|l'(\tau)| \neq 0$. It follows then $a = -\infty$ and $b = \infty$, and hence the threshold in* (6) *is trivially 0, for any target model $\mathbf{w}$. Of course, our characterization in Theorem 2 continues to hold for any domain $\mathbb{Z}$, unbounded or not.*

**Example 4** (Dichotomy)**.** *Consider the smooth loss*[8]

$$l(t) = \begin{cases} -(4t+1)\exp(-2), & \text{if } t \leq -\frac{1}{2} \\ \exp(\frac{1}{t}), & \text{if } t \in [-\frac{1}{2}, 0] \\ 0, & \text{if } t \geq 0 \end{cases}. \tag{A.8}$$

*Clearly, we have $a = 0$ and $b = \infty$. Thus, we arrive at a remarkable dichotomy:*

- *If $\langle \mathbf{w}, \mathbf{g}(\mu) \rangle = 0$ (in particular any separating $\mathbf{w}$), then data poisoning succeeds with any $\lambda > 0$;*
- *If $\langle \mathbf{w}, \mathbf{g}(\mu) \rangle \neq 0$ (and hence $\mathbf{w}$ cannot separate $\mu$), then data poisoning fails with any $\lambda < 1$.*

Note that $l$ in (A.8) is not calibrated since $l'(0) = 0$ (Bartlett et al. 2006), so it may not be a sensible loss to use in practice. For a calibrated margin loss $l$, i.e., one that is differentiable at 0 with $l'(0) < 0$, we necessarily have $b > 0$ and $a < 0$, so the threshold in (6) usually lies strictly in $(0, 1)$, incurring a nontrivial phase transitioning.

**Theorem 6** (Multiclass)**.** *Consider $\ell(\mathbf{x}, \mathbf{y}; W) = l(W^\top \mathbf{x}, \mathbf{y})$ for some loss $l$. Then*[9]*,*

$$G(\mathbf{x}, \mathbf{y}) := \nabla_W \ell(\mathbf{x}, \mathbf{y}; W) = \mathbf{x} \otimes \nabla l(W^\top \mathbf{x}, \mathbf{y}), \tag{A.9}$$

*and $W$ is $\lambda$-poisoning reachable iff there exists $\nu \in \Gamma$ such that*

$$\mathbf{0} \in (1 - \lambda) G(\mu) + \lambda G(\nu). \tag{A.10}$$

*Suppose $W^\top G(\mu)$ is non-degenerate and let*

$$\mathbb{J} := \{\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\nu}(W^\top \mathbf{x}) \otimes \nabla l(W^\top \mathbf{x}, \mathbf{y}) : \nu \in \Pi\}.$$

*Then, $W$ is $\lambda$-poisoning reachable if $\Gamma \supseteq T_{\#}\Pi$ and*

$$\mathbf{0} \in (1 - \lambda) W^\top G(\mu) + \lambda \mathbb{J}, \tag{A.11}$$

*where the transformation $T(\mathbf{x}, \mathbf{y}) := \big(G(\mu)[W^\top G(\mu)]^{-1} W^\top \mathbf{x}, \mathbf{y}\big)$. Conversely,* (A.11) *holds if $W$ is $\lambda$-poisoning reachable and $\Pi \supseteq \Gamma$.*

*Proof.* The proof is completely similar to that of Theorem 5. □

*Proof.* [of Example 3] We aim to show that for any $\mathbf{h} \in \mathbb{R}^c$ and one-hot $\mathbf{y} \in \mathbb{R}^c$, we have

$$-\mathscr{W}(\tfrac{c-1}{e}) \leq \langle \mathbf{h}, \mathbf{p} - \mathbf{y} \rangle \leq \infty, \text{ where recall that } \mathbf{p} := \operatorname{softmax}(\mathbf{h}) = \exp(\mathbf{h}) / \sum_k \exp(h_k). \tag{A.12}$$

[8] This is essentially a smoothed version of the perceptron loss $l(t) = \max\{-t, 0\}$.

[9] We use the notation $\mathbf{a} \otimes \mathbf{b} := \mathbf{a}\mathbf{b}^\top$ for two column vectors.

The right-hand side is clear: we need only send some $h_k$ to $\infty$, as long as $y_k \neq 1$. For the left-hand side, we simplify as follows. W.l.o.g. assume $y_i = 1$. Then,

$$\langle \mathbf{h}, \mathbf{p} - \mathbf{y} \rangle = \sum_k h_k \left[ \frac{\exp(h_k)}{\sum_j \exp(h_j)} - y_k \right] = \frac{\sum_k (h_k - h_i) \exp(h_k - h_i)}{1 + \sum_{j \neq i} \exp(h_j - h_i)} \tag{A.13}$$

$$= \sum_{k \neq i} \frac{\frac{1}{c-1} + \exp(h_k - h_i)}{1 + \sum_{j \neq i} \exp(h_j - h_i)} \cdot \frac{(h_k - h_i) \exp(h_k - h_i)}{\frac{1}{c-1} + \exp(h_k - h_i)} \tag{A.14}$$

$$\geq \inf_t \frac{t \exp(t)}{\frac{1}{c-1} + \exp(t)} \tag{A.15}$$

$$= -\mathscr{W}(\tfrac{c-1}{e}), \tag{A.16}$$

where the inequality is achieved when $t \equiv h_k - h_i$ minimizes (A.15). □

**Theorem 4** (Neural Networks)**.** *Consider $\ell(\mathbf{x}, \mathbf{y}; W, \mathbf{u}) = l(\mathbf{h}, \mathbf{y})$ for some loss $l$, where $\mathbf{h} := W^\top \varphi(\mathbf{x}; \mathbf{u})$. Then,*

$$\nabla_W \ell(\mathbf{x}, \mathbf{y}; W, \mathbf{u}) = \varphi(\mathbf{x}; \mathbf{u}) \otimes \nabla_{\mathbf{h}} l(\mathbf{h}, \mathbf{y}) \tag{11}$$

$$\nabla_{\mathbf{u}} \ell(\mathbf{x}, \mathbf{y}; W, \mathbf{u}) = \nabla_{\mathbf{u}} \varphi(\mathbf{x}; \mathbf{u}) W \nabla_{\mathbf{h}} l(\mathbf{h}, \mathbf{y}), \tag{12}$$

*and $(W, \mathbf{u})$ is $\lambda$-poisoning reachable iff there exists $\nu \in \Gamma$ such that*

$$\mathbf{0} \in (1 - \lambda) G(\mu) + \lambda G(\nu), \tag{13}$$

*where $G(\nu) := \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \nu} (\nabla_W \ell, \nabla_{\mathbf{u}} \ell)$. In particular, $(W, \mathbf{u})$ is $\lambda$-poisoning reachable only if there exists some $\nu \in \Gamma$ such that*

$$\mathbf{0} \in (1 - \lambda) G_1(\mu) + \lambda G_1(\nu), \tag{14}$$

*where $G_1(\nu) := \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \nu} \varphi(\mathbf{x}; \mathbf{u}) \otimes \nabla_{\mathbf{h}} l(\mathbf{h}, \mathbf{y})$.*

*Proof.* It is straightforward to compute the gradients in (11) and (12). The iff condition in (13) then follows from Theorem 1. The necessary condition in (14) is obtained by simply ignoring the second part of $G(\mu)$ (that corresponds to $\nabla_{\mathbf{u}} \ell$). □

From (14) we conclude that the poisoning distribution $\nu$ must be supported at least on $s = \operatorname{rank}(G_1(\mu))$ points, as long as $\lambda \in (0, 1)$. Taking inner product w.r.t. $WA$ on both sides of (14) we obtain

$$0 = (1 - \lambda) g_A(\mu) + \lambda g_A(\nu), \tag{A.17}$$

where $g_A(\nu) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \nu} \langle A \nabla_{\mathbf{h}} l(\mathbf{h}, \mathbf{y}), \mathbf{h} \rangle$ and $A$ is arbitrary. The condition (A.17) is univariate and easy to check, albeit being necessary but not sufficient. We remark that the free choice of the matrix $A$ may be exploited to tighten this necessary condition.

## B. Data poisoning as measure optimization

In this section we discuss a measure optimization approach for solving the gradient canceling problem:

$$\min_{\nu \in \Gamma} \ \tfrac{1}{2} \|\mathbf{g}(\mu) + \varepsilon_d \mathbf{g}(\nu)\|_2^2, \tag{B.1}$$

where we recall that

$$\mathbf{g}(\nu) = \mathbb{E}_{\mathbf{z} \sim \nu} \nabla_{\mathbf{w}} \ell(\mathbf{z}; \mathbf{w}) \tag{B.2}$$

is the model gradient computed over the distribution $\nu$. The objective of (B.1) is a convex quadratic, although living in an infinite dimensional space (the vector space of all signed measures over $\mathbb{Z}$). A particularly suitable way to solve (B.1) is the well-known Frank-Wolfe algorithm, where we repeatedly perform "atomic" updates to the measure $\nu$:

$$\nu_{t+1} \leftarrow (1 - \eta_t) \nu_t + \eta_t \zeta_t, \tag{B.3}$$

where $\eta_t$ is the step size, e.g., $\eta_t = \frac{2}{t+2}$. The direction $\zeta_t$ is found by solving the linear minimization subproblem:

$$\min_{\zeta \in \Gamma} \langle \mathbf{g}(\mu) + \varepsilon_d \mathbf{g}(\nu_t), \mathbf{g}(\zeta) \rangle \,. \tag{B.4}$$

When $\Gamma = \mathcal{P}$ consists of all distributions over $\mathbb{Z}$, the above subproblem simplifies to:

$$\min_{\mathbf{z} \in \mathbb{Z}} \langle \mathbf{g}(\mu) + \varepsilon_d \mathbf{g}(\nu_t), \nabla_{\mathbf{w}} \ell(\mathbf{z}; \mathbf{w}) \rangle \,, \tag{B.5}$$

i.e., we find a new poisoning point $\mathbf{z}$ to add to the support of the poisoning distribution $\nu_t$, while the step (B.3) adjusts the probability mass. One particularly appealing part of this algorithm is that after $t$ iterations, the candidate poisoning distribution $\nu_t$ is supported at most on $t+1$ points (assuming we start with some $\nu_0$ supported on a single point). We remark that the subproblem (B.5) is often nonconvex (in particular for neural networks), and could be challenging to solve. The other difficulty is that an attacker often is not allowed to upload an entire poisoning distribution, so a resampling procedure (on $\nu$) will be necessary to create a poisoning set, which is why we opted for a more direct approach in the main paper.

Another possibility is to parameterize $\nu$ as the push-forward of some fixed distribution (e.g., the training distribution), i.e., $\nu = [T(\theta)]_{\#}\mu$, and we optimize the push-forward transformation $T(\theta)$.

## C. Additional Experiments

### C.1. Additional implementation details

**Hardware and package:** experiments were run on a cluster with `T4` and `P100` GPUs. The platform we use is PyTorch (Paszke et al. 2019).

**Model in details**: for the MNIST dataset, we examine three target models: Logistic Regression; a neural network (NN) with three layers, where we choose hidden size as 784 and apply leaky ReLU with `negative_slope` $= 0.2$ for activation; and a convolutional neural network (CNN) with two convolutional layers with kernel size 3, maxpooling and two fully connected layers with hidden size 128.

**Synthetic Datasets:** in Figure 1 and Figure 2, we perform experiemnts on two synthetic datasets. (1) OR dataset: we simply use the OR dataset in 2D space in Figure 4 and repeat each point for 50 times (200 samples in total) with small Gaussian noise. (2) 10-D Gaussian dataset: we use the `make_classification` function in `sklearn.datasets` with 1000 samples and 10 features.

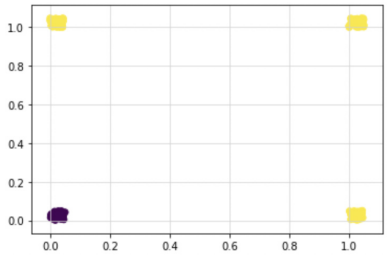


Figure 4: Here we visualize the OR dataset.

**More on GradPC**: to choose proper target parameters (specifically, $\varepsilon_w$), we use validation sets described in Section 5 for accuracy drop comparison. The GradPC attack never sees the test set during the construction of its perturbed models.

**Batch size:** for the Gradient Canceling experiments on MNIST, we set batch size as the size of entire training set (60000) for simplicity. For CIFAR-10 and TinyImageNet experiments, we set batch size as 1000 due to CUDA memory size constraint.

**Optimizer, learning rate scheduler and hyperparameters:** we use SGD with momentum for optimization and the cosine learning rate scheduler (Loshchilov and Hutter 2017) for the Gradient Canceling algorithm. We set the initial learning rate as 0.5 and run 1000 epochs across every experiment.

### C.2. More on Parameter Corruption

Recall in Table 1 we compare TGDA with GradPC briefly. Here we show the complete results with more choices of $\varepsilon_w$ and an additional baseline method called RandomPC in Sun et al. (2020) in Table 5.

Table 5: The attack accuracy/accuracy drop (%) on the MNIST dataset.

| Target Model | Clean | TGDA | RandomPC | | | GradPC | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | Accuracy/Drop | $\varepsilon_w = 0.01$ | $\varepsilon_w = 0.1$ | $\varepsilon_w = 1$ | $\varepsilon_w = 0.01$ | $\varepsilon_w = 0.1$ | $\varepsilon_w = 1$ |
| LR | 92.35 | 89.56 / 2.79 ($\varepsilon_w = 2.45$) | 91.94 / 0.41 | 81.24 / 11.11 | 24.66 / 67.69 | 91.91 / 0.44 | 89.72 / 2.63 | **21.48 / 70.87** |
| NN | 98.04 | 96.54 / 1.50 ($\varepsilon_w = 0.55$) | 97.62 / 0.42 | 82.67 / 15.37 | 32.77 / 65.27 | 97.63 / 0.41 | 97.05 / 0.99 | **31.14 / 66.90** |
| CNN | 99.13 | 98.02 / 1.11 ($\varepsilon_w = 0.74$) | 98.84 / 0.29 | 72.00 / 27.13 | 19.26 / 79.87 | 98.74 / 0.39 | 98.69 / 0.44 | **12.98 / 86.15** |

### C.3. Least-square Regression

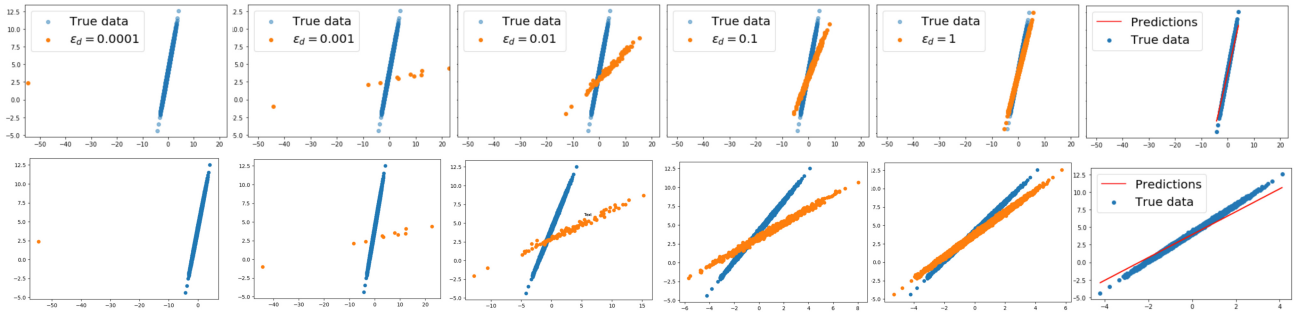


Figure 5: Here we run the Gradient Canceling algorithm on linear regression on a 2D Gaussian dataset. The first row displays all figures in the same scale for better comparison; and the second row shows the upper figures in their original scale for better viewing. (1) The left five figures show the poisoned points generated with different $\varepsilon_d$. When $\varepsilon_d$ is smaller, the poisoned points are farther from the data distribution. (2) The algorithm always generates the target parameter (the prediction) regardless of $\varepsilon_d$.

Recall that from Example 1 we conclude data poisoning with any $\varepsilon_d > 0$ is possible for least-square regression. We perform GC attack on a synthetic 2D Gaussian dataset and visualize the results in Figure 5. We observe that (1) the algorithm always generates the target parameter regardless of $\varepsilon_d$, which immediately verifies our conclusion; (2) by increasing $\varepsilon_d$, the poison distribution $\nu$ gradually moves towards the data distribution $\mu$, which makes intuitive sense. Moreover, recall that we may restrict the search of a poisoning distribution to the potentially much smaller subset $T_{\#}\Pi$ (where $\mathbf{x}$ lies on the line spanned by $\mathbf{g}(\mu)$), while in practice GC does not seem to always follow this theoretical construct.

### C.4. Comparison with Gradient Matching

As we mentioned in Section 4, one of the difference between Gradient Matching and our work is that there is no guarantee that after retraining over $\hat{\nu}$, gradient matching will arrive at the target model while our Algorithm 1 explicitly aims to achieve this goal. We have shown in the Figure 5 that GC empirically achieve the target model regardless of $\varepsilon_d$. By comparing with Figure 6, we observe that gradient matching achieves different model parameters for every $\varepsilon_d$.

### C.5. More on Figure 3

Recall that in Figure 3, we fix $\varepsilon_d$ and draw the learning curve for GC optimization for different $\varepsilon_w$, where the $y$-axis indicates the normalized loss, i.e., $\|\mathbf{g}(\chi)\|$. We observe that when $\tau > \varepsilon_d$, $\|\mathbf{g}(\chi)\|$ converges to a larger value, influenced by the distance between $\tau$ and $\varepsilon_d$.

Conversely, we fix $\varepsilon_w$ (consequently, $\tau$) for different target models and repeat the MNIST experiments. In Figure 7, we again observe that when $\tau > \varepsilon_d$, $\|\mathbf{g}(\chi)\|$ converges at a relatively bigger number. Overall, we have confirmed the theoretical limitations proved in Section 3.

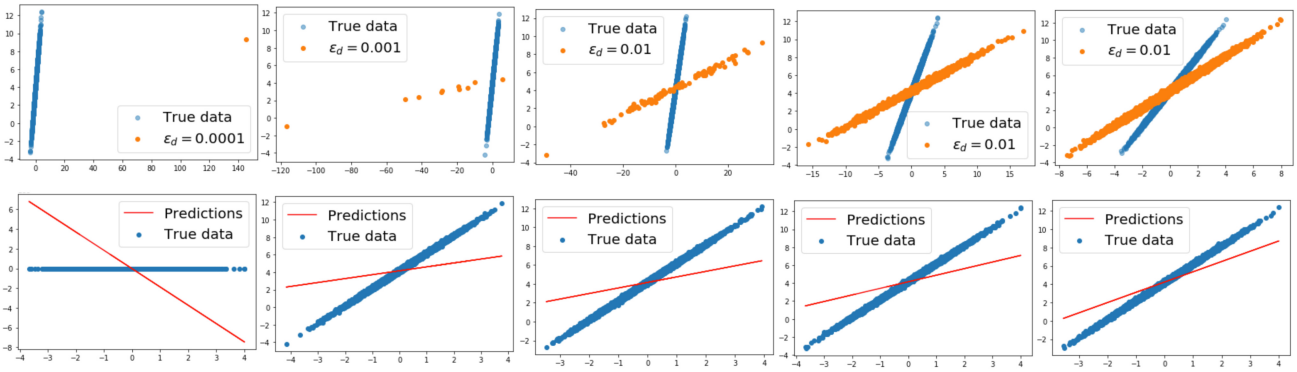


Figure 6: Here we run the Gradient Matching algorithm on linear regression on a 2D Gaussian dataset. The first row displays the poisoned points generated with different $\varepsilon_d$; and the second row shows the different target parameters generated by the algorithm with different $\varepsilon_d$.

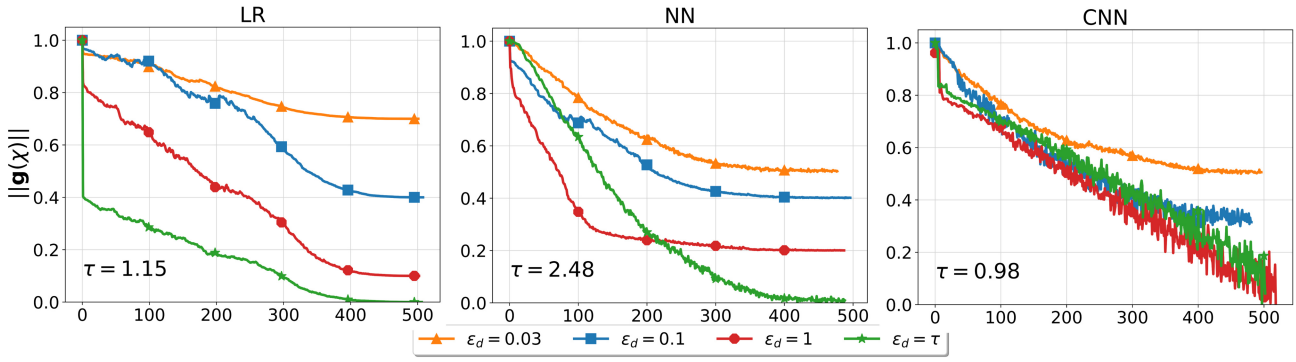


Figure 7: The learning curve for running GC on MNIST with different target models and $\varepsilon_d$. Note that we fix $\varepsilon_w$ for each model and print the respective $\tau$, and the loss indicates $\|\mathbf{g}(\chi)\|$. The figure again confirms that GC cannot achieve $\mathbf{w}$ if $\varepsilon_d < \tau$.

### C.6. Scaling $\mathbf{w}$

We point out a subtlety in Example 2: by scaling $\mathbf{w}$ towards the origin, we do not change its accuracy (except the confidence it induces). However, the threshold $\tau$ tends to 0 and hence data poisoning succeeds in producing the target parameter $\mathbf{w}$ with a smaller $\lambda$. In other words, less confident models are easier to poison to, which makes intuitive sense. For verification, we run the GC attack with scaled target parameter $\mathbf{w}/2$ and compare it with the original target parameter $\mathbf{w}$ in Table 6. With the same target model accuracy, scaling $\mathbf{w}$ significantly reduces its corresponding $\tau$, making it easier to poison to.

Table 6: The GC attack accuracy drop (%) on MNIST when scaling $\mathbf{w}$ by half.

| **Target Model** | clean | GradPC | $\tau(\mathbf{w})$ | $\tau(\mathbf{w}/2)$ | $\varepsilon_d$ | $\mathbf{w}$ | $\mathbf{w}/2$ |
|---|---|---|---|---|---|---|---|
| LR | 92.35 | -70.87 | 1.15 | 0.54 | 0.03 | -22.97 | -44.11 |
| | | | | | 0.1 | -63.83 | -67.22 |
| | | | | | 1 | -67.01 | -79.55 |
| NN | 98.04 | -20.03 | 2.48 | 1.41 | 0.03 | -6.10 | -9.29 |
| | | | | | 0.1 | -9.77 | -11.01 |
| | | | | | 1 | -12.05 | -15.33 |
| CNN | 99.13 | -24.78 | 0.98 | 0.42 | 0.03 | -9.55 | -12.03 |
| | | | | | 0.1 | -20.10 | -21.55 |
| | | | | | 1 | -23.80 | -24.56 |

The fact that simply scaling $\mathbf{w}$ down could improve its poisoning reachability might seem surprising at first glance. However, this is due to a mismatch between how we train and how we test. It is best to explain this observation in the binary setting, where we note the mismatch between the common prediction rule

$$\hat{y} = \operatorname{sign}(\langle \mathbf{x}, \mathbf{w} \rangle), \tag{C.1}$$

which is invariant to (positive) scaling (of $\mathbf{w}$), and our training objective in finding a good parameter $\mathbf{w}$, e.g., through logistic regression:

$$\inf_{\mathbf{w}} \ \frac{1}{n} \sum_i \log[1 + \exp(-y_i \langle \mathbf{x}_i, \mathbf{w} \rangle)], \tag{C.2}$$

which is not invariant to scaling (of $\mathbf{w}$).

Let us give an explicit example to further demonstrate this point. Consider three (cleaning) training points[10] on the real line:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, y_1 = +; \quad \mathbf{x}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, y_2 = +; \quad \mathbf{x}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, y_3 = -; \tag{C.3}$$

where we have padded 1 at the last entry of each $\mathbf{x}$ (so that we can absorb the bias $b$ into $\mathbf{w}$). Setting the derivative of (C.2) w.r.t. $\mathbf{w}$ to zero we obtain:

$$3 \cdot \mathbf{g}(\mathbf{w}) = -\frac{1}{1 + \exp(w_1 + w_2)} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \frac{1}{1 + \exp(w_2 - w_1)} \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \frac{1}{1 + \exp(-w_2)} \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \mathbf{0}. \tag{C.4}$$

Solving the above equation we have $\mathbf{w}_\star = \begin{bmatrix} 0 \\ \ln 2 \end{bmatrix}$.

Now consider the scenario where we are given a target parameter $\mathbf{w} = 2\mathbf{w}_\star$. According to our theory, the poisoning ratio

$$\varepsilon_d > \tau :\approx \max \left\{ 1.2 \left[ \frac{-w_1 - w_2}{1 + \exp(w_1 + w_2)} + \frac{w_1 - w_2}{1 + \exp(w_2 - w_1)} + \frac{w_2}{1 + \exp(-w_2)} \right], 0 \right\} \tag{C.5}$$

$$= \max \left\{ 1.2 w_2 \cdot \frac{\exp(w_2) - 2}{1 + \exp(w_2)}, 0 \right\} \tag{C.6}$$

$$= 0.48 \cdot 2 \ln 2 \approx 0.67. \tag{C.7}$$

[10] This is in fact the smallest example: with 2 or fewer training points, logistic regression does not attain the infimum.

In other words, the poisoning set needs to be as large as 67% of the training set, in order to produce $\mathbf{w} = 2\mathbf{w}_\star$. However, if we scale $\mathbf{w}$ down to $\mathbf{w}_\star$, then we do not even need to add any poisoned point, since $\mathbf{w}_\star$ is already stationary (by definition). If we continue to scale $\mathbf{w}$ down to say $0.5\mathbf{w}_\star$ (so that $\langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle < 0$ and hence $\tau = 0$), then for any $\varepsilon > 0$, we may put $\varepsilon$ copies of $\mathbf{x} = \alpha \mathbf{g}(\mathbf{w}) \approx \frac{1}{\varepsilon}\mathbf{g}(\mathbf{w})$ as the poisoning set to produce $\mathbf{w}$; see the detailed analysis below.

More generally, consider adding $\varepsilon$ copies of a poisoning data point at $\mathbf{x} = \alpha \mathbf{g}(\mathbf{w}), y = +$ (where $\alpha \in \mathbb{R}$ will be determined later) so that the gradient on the clean and poisoned data is proportional to:

$$\mathbf{g}(\mathbf{w}) - \varepsilon \frac{1}{1 + \exp(\langle \mathbf{w}, \mathbf{x} \rangle)} \mathbf{x} = \mathbf{g}(\mathbf{w}) - \varepsilon \frac{1}{1 + \exp(\langle \mathbf{w}, \alpha \mathbf{g}(\mathbf{w}) \rangle)} \alpha \mathbf{g}(\mathbf{w}) = \mathbf{g}(\mathbf{w}) \cdot \left[ 1 - \varepsilon \frac{\alpha}{1 + \exp(\alpha \langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle)} \right]. \tag{C.8}$$

We can break the analysis into a few cases now:

- $\mathbf{g}(\mathbf{w}) = \mathbf{0}$, i.e., we scale $\mathbf{w}$ down to $\mathbf{w}_\star$, in which case no poisoning point is needed to produce $\mathbf{w} = \mathbf{w}_\star$.
- $\langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle = 0$, in which case the gradient reduces to $\mathbf{g}(\mathbf{w})[1 - \varepsilon\alpha/2]$. Therefore, for any $\varepsilon > 0$, we may produce $\mathbf{w}$ by putting $\varepsilon$ copies of $\mathbf{x} = \frac{2}{\varepsilon}\mathbf{g}(\mathbf{w})$.
- $\langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle < 0$, in which case for any $\varepsilon > 0$, the function

  $$\alpha \mapsto 1 + \exp(\alpha \langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle - \varepsilon\alpha \tag{C.9}$$

  clearly has a zero $\alpha_*$ (easily seen by letting $\alpha \to \pm\infty$ and applying the intermediate value theorem). Thus again, we may produce $\mathbf{w}$ by putting $\varepsilon$ copies of $\mathbf{x}$ at $\alpha_* \mathbf{g}(\mathbf{w})$. (Note that $\alpha_* \to \infty$ if $\varepsilon \to 0$; roughly $\alpha_* \approx \frac{1}{\varepsilon}$.)
- $\langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle > 0$, in which case the function

  $$\alpha \mapsto 1 + \exp(\alpha \langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle - \varepsilon\alpha \tag{C.10}$$

  has a zero $\alpha_*$ iff $\varepsilon \geq \tau$. Indeed,

  $$1 = \varepsilon \frac{\alpha}{1 + \exp(\alpha \langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle)} \iff 1 = \frac{\varepsilon}{\langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle} \cdot \frac{\alpha \langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle}{1 + \exp(\alpha \langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle)} \tag{C.11}$$

  $$\leq \frac{\varepsilon}{\langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle} \cdot \sup_t \frac{t}{1 + \exp(t)} \tag{C.12}$$

  $$= \frac{\varepsilon}{\langle \mathbf{w}, \mathbf{g}(\mathbf{w}) \rangle} \cdot \mathcal{W}(1/e). \tag{C.13}$$

  Thus again, for any $\varepsilon \geq \tau$, we may produce $\mathbf{w}$ by putting $\varepsilon$ copies of $\mathbf{x}$ at $\alpha_* \mathbf{g}(\mathbf{w})$.

In Figure 8 we observe that GC converged to *nonzero* loss (i.e., unable to produce the target parameter) when $\varepsilon_d < \tau$, for any learning rate we tried, while after scaling $\mathbf{w}$ down so that $\varepsilon_d > \tau$, GC immediately converged to zero loss (without the need of tuning the learning rate), confirming our theoretical analysis above. We plan to further explore the scaling effect in future work.

### C.7. Simulating Different Target Parameters

Next, we verify if GC can achieve any desired target parameter. We choose poisoned models generated by TGDA attack as target parameters and perform PC. We discover that such parameters are also achievable by GC in Table 7, which further confirms that GC may be equipped with any other parameter corruption methods, regardless of how the target parameters are generated.

### C.8. GC against Defenses

Next, we examine the GC attack against three popular distribution-wise defenses. (1) Influence defense (Koh and Liang 2017) removes $\varepsilon_d$ suspicious points according to higher influence functions; (2) Sever (Diakonikolas et al. 2019) removes $\varepsilon_d$ training points with the highest outlier scores, defined using the top singular value in the matrix of gradients; (3) Maxup defense (Gong et al. 2021) generates a set of augmented data with random perturbations and then aims at minimizing the worst case loss over the augmented data.

We present our results on the MNIST dataset in Table 8 and observe that: (1) Among the three defenses, Sever is the most effective one and can significantly reduce the effectiveness of GC. (2) Clipping the poisoned data to the range of clean

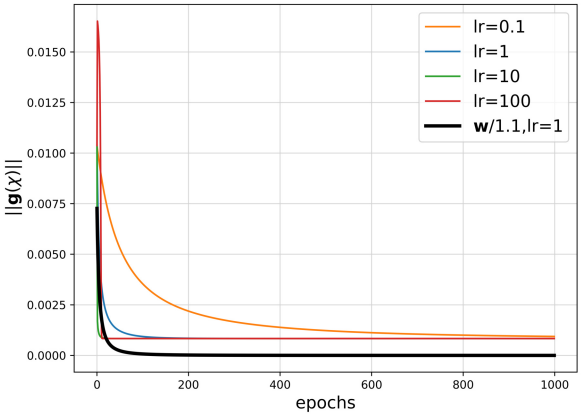


Figure 8: We plot the training curves of GC on the toy example (see (C.3)). The colored curves represent GC attack on $\mathbf{w} = \begin{bmatrix} 0 \\ 2\ln 2 \end{bmatrix}$ with $\varepsilon = 0.52 < \tau \approx 0.67$ under different learning rates; the black curve represents GC attack on the scaled target parameter $\mathbf{w}/1.1 \approx \begin{bmatrix} 0 \\ 1.82\ln 2 \end{bmatrix}$ with $\varepsilon = 0.52 > \tau \approx 0.51$ under learning rate 1.

Table 7: Simulating TGDA attack ($\varepsilon_d = 1$) with Gradient Canceling attack on the MNIST dataset.

| **Target Model** | clean | TGDA | $\tau$ | $\varepsilon_d$ | **GC** |
|---|---|---|---|---|---|
| LR | 92.35 | -8.97 | 2.33 | 0.03 | -2.66 |
| | | | | 0.1 | -3.39 |
| | | | | 1 | -5.53 |
| | | | | $\tau$ | -8.35 |
| NN | 98.04 | -5.49 | 0.95 | 0.03 | -1.39 |
| | | | | 0.1 | -1.55 |
| | | | | 1 | -4.99 |
| CNN | 99.13 | -4.76 | 0.49 | 0.03 | -0.98 |
| | | | | 0.1 | -2.10 |
| | | | | 1 | -4.68 |

training set makes GC more robust against all defenses, with the tradeoff of attack effectiveness. (3) Larger $\varepsilon_d$ makes the attack generally more robust, which matches our observation in least-squared regression.

### C.9. Visualization of Poisoned Images

Finally, we visualize some poisoned images generated by the GC attack in Figure 9 and Figure 10.

Table 8: The accuracy drop (%) Gradient Canceling attack (w/wo clipping) introduces on MNIST with Influence/Sever/MaxUp defense (+ indicates the accuracy increased by defenses). GC: original Gradient Canceling attack; GC-c: GC with clipped output; GC-d: GC after defense; GC-cd: GC-c after defense.

| **Model** | Clean Acc | $\varepsilon_d$ | GC | GC-c | Influence | | Sever | | MaxUp | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | GC-d | GC-cd | GC-d | GC-cd | GC-d | GC-cd |
| LR | 92.35 | 0.03 | -22.79 | -11.28 | -21.99 / +0.80 | -11.17 / +0.11 | -12.81 / **+9.98** | -9.66 / **+1.62** | -22.59 / +0.20 | -11.26 / +0.02 |
| | | 0.1 | -63.83 | -26.77 | -63.51 / +0.32 | -26.67 / +0.10 | -59.79 / **+4.04** | -25.53 / **+1.24** | -63.65 / +0.18 | -26.67 / +0.10 |
| | | 1 | -67.01 | -28.99 | -66.75 / +0.26 | -26.71 / +0.06 | -65.01 / **+2.00** | -27.89 / **+1.10** | -66.02 / +0.09 | -28.97 / +0.02 |
| NN | 98.04 | 0.03 | -6.10 | -3.25 | -5.59 / +0.51 | -3.16 / +0.09 | -3.22 / **+2.88** | -2.26 / **+0.90** | -6.08 / +0.02 | -3.24 / +0.01 |
| | | 0.1 | -9.77 | -5.10 | -9.32 / +0.45 | -5.02 / +0.08 | -7.66 / **+2.11** | -4.46 / **+0.56** | -9.76 / +0.01 | -5.10 / +0.00 |
| | | 1 | -12.05 | -6.53 | -11.65 / +0.40 | -6.48 / +0.05 | -10.02 / **+2.03** | -6.11 / **+0.42** | -12.04 / +0.01 | -6.53 / +0.00 |
| CNN | 99.13 | 0.03 | -9.55 | -5.87 | -8.57 / +0.98 | -5.56 / +0.31 | -5.55 / **+4.00** | -4.36 / **+1.51** | -9.39 / +0.16 | -5.83 / +0.04 |
| | | 0.1 | -20.10 | -12.50 | -19.19 / +0.91 | -12.35 / +0.15 | -16.55 / **+3.55** | -11.32 / **+1.18** | -20.06 / +0.04 | -12.48 / +0.02 |
| | | 1 | -23.80 | -13.32 | -23.10 / +0.70 | -13.21 / +0.11 | -21.05 / **+2.75** | -12.51 / **+0.81** | -23.79 / +0.01 | -13.32 / +0.00 |

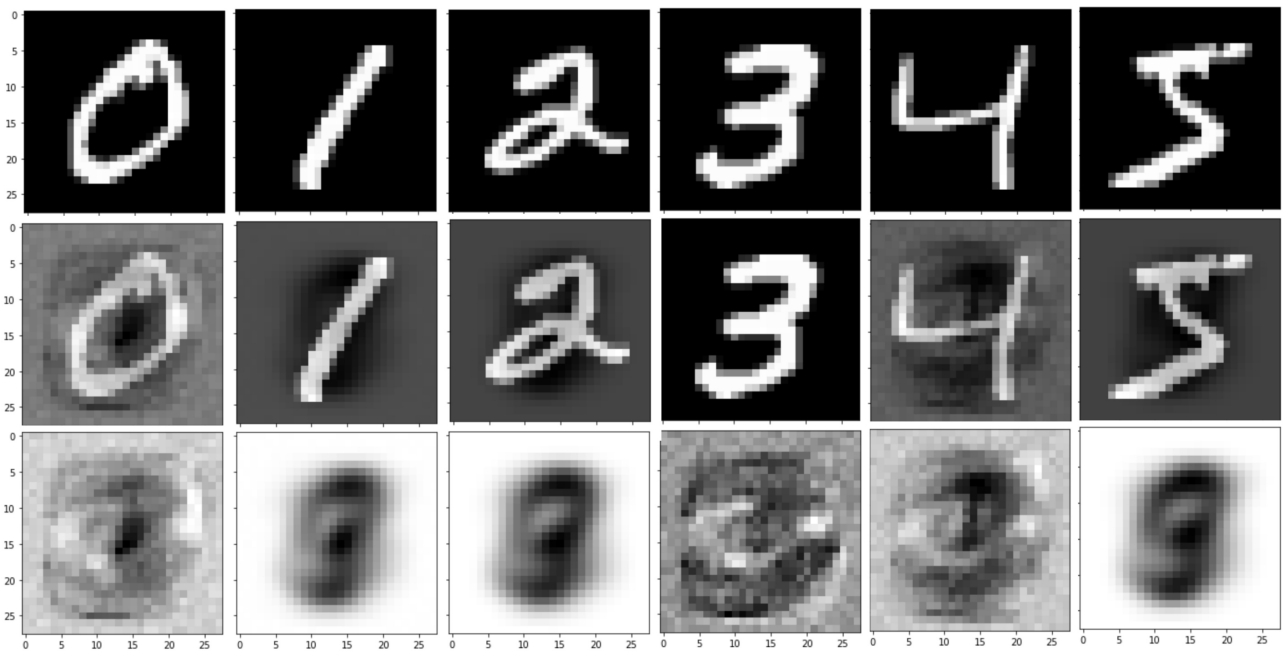

Figure 9: We visualize some poisoned images generated by the GC attack on the MNIST dataset. The first row shows the clean samples, the second row shows the poisoned samples; the third row displays the perturbation.

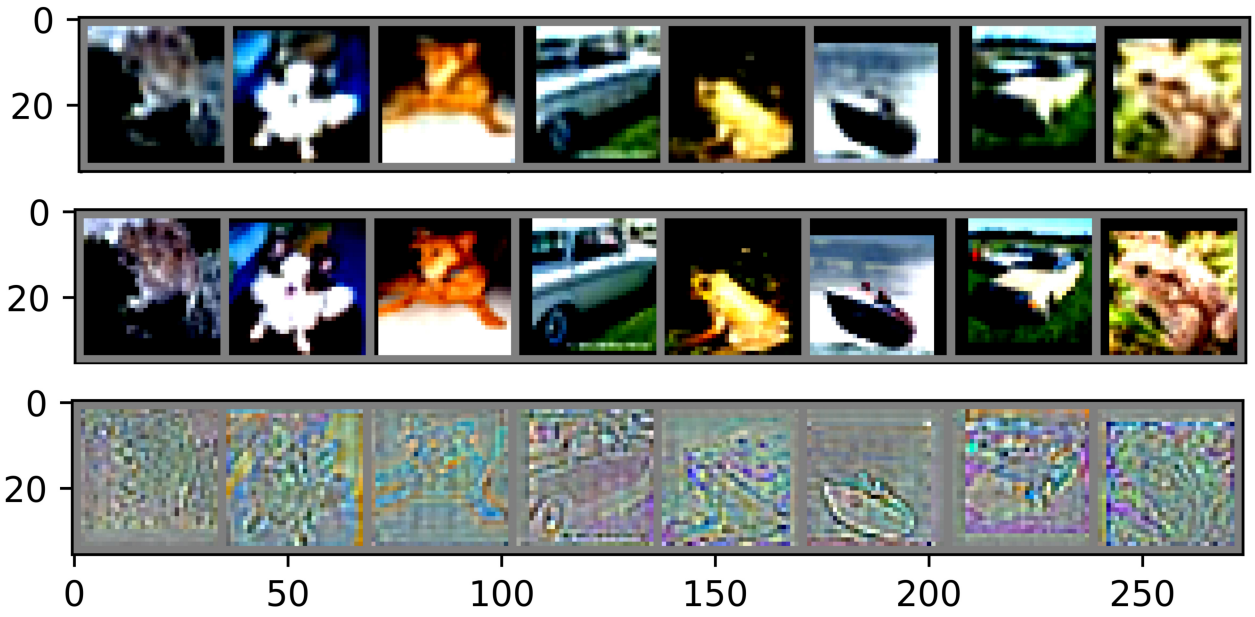


Figure 10: We visualize some poisoned images generated by the GC attack on the CIFAR-10 dataset. The first row shows the clean samples, the second row shows the poisoned samples; the third row displays the perturbation.

### C.10. Comparison with Replacing Attack

In this work we only consider an adversary who is restricted to *add* corrupted points $\mathcal{D}_p$ to the intact (clean) training set $\mathcal{D}_{tr}$, while an even stronger attacker might consider replacing part of $\mathcal{D}_{tr}$ with $\mathcal{D}_p$ (also closely related to the nasty noise model Bshouty et al. 2002). We first formulate the general case: recall that we consider the mixed distribution $\chi = (1-\lambda)\mu + \lambda\nu$ of the clean distribution $\mu$ and poisoned distribution $\nu$, where $\lambda$ is the proportion of poisoning data. Then, replacing part of the clean training data is equivalent to:

$$\chi = (1-\lambda')\mu' + \lambda'\nu, \tag{C.14}$$

where $\mu'$ is a subset of $\mu$, and $\lambda' = |\mathcal{D}_p|/|\mathcal{D}_{tr}|$. Empirically, with the ability to replace data points we may still apply Gradient Canceling in a straightforward manner: the only difference is that in Algorithm 1 we change $\mu$ to $\mu'$, a random subset of $\mu$. Following this idea, we perform a simple experiment: we choose $\varepsilon_d = 0.03$, and choose $\mu'$ to be a random subset of $\mathcal{D}_{tr}$, with size $\frac{1}{1+\varepsilon_d}|\mathcal{D}_{tr}| \approx 0.97|\mathcal{D}_{tr}|$. The results on MNIST are presented below in Table 9:

Table 9: Gradient Canceling attack ($\varepsilon_d = 0.03$) with adding-only vs replacing-only on the MNIST dataset.

| **Target Model** | clean | **GC (adding-only)** | **GC (replacing)** |
|---|---|---|---|
| LR | 92.35 | -22.97 | -23.10 |
| NN | 98.04 | -6.10 | -6.35 |
| CNN | 99.13 | -9.55 | -9.62 |

We observe that the ability to replace clean training data is indeed (slightly) more powerful than the corresponding adding-only attack. Notably, we remove training samples randomly, which may not be relatively weak. Ideally, an adversary would remove the most important points (e.g., in Ilyas et al. 2022) to further reduce the test accuracy. This improved replacing attack might be worth future exploration, although we note that it is less likely to be applicable when an attacker does not have direct access to a victim's infrastructure.

## D. Selecting Target Parameters

Here we discuss how to select an appropriate target parameter $\mathbf{w}$ for the GC attack. In principle, there are two major factors regarding the selection of target parameters $\mathbf{w}$: (1) strength of $\mathbf{w}$, measured by the test accuracy drop it incurs; (2) poisoning reachability, measured by the optimality condition (i.e., the empirical loss in Equation (18)). We want to choose a $\mathbf{w}$ that is both reachable and as strong as possible. Next, we discuss both criteria in details:

- **Strength of $\mathbf{w}$**: (a) existing works (e.g., Koh et al. 2022; Suya et al. 2021) only explored rudimentary ways to construct target parameters (e.g., through the label flip attack), and thus are less effective in casting particularly powerful target parameters; (b) with GradPC, we can now easily quantify the strength of a target parameter $\mathbf{w}$ using $\varepsilon_w$ (in Table 1); (c) thus in practice, we first prepare a sequence of target parameters $\{\mathbf{w}_k : k \in K\}$ with $|K|$ different $\varepsilon_w$, and then send them all to the reachability test (in the next step).
- **Reachability test**: (a) given the list of $\{\mathbf{w}_k : k \in K\}$, we first calculate every corresponding $\tau(\mathbf{w}_k)$, such that we can already rule out a few choices where $\varepsilon_d < \tau$ (that we know GC cannot achieve with the existing budget $\varepsilon_d$). After this process, we only keep a subset of target parameters $\{\mathbf{w}_k : k \in \bar{K}\}$; (b) next, we run GC for each $\{\mathbf{w}_k : k \in \bar{K}\}$, and examine if GC can achieve them by checking the loss upon convergence. We only keep those $\mathbf{w}_k$'s that return a loss smaller than a margin (this margin is defined by one-tenth of the initial loss) when $\varepsilon_d \approx \tau$. (c) finally, we empirically select a target parameter $\mathbf{w}$ with the largest accuracy drop on the validation set.

## Extra References

Bartlett, P. L., M. I. Jordan, and J. D. McAuliffe (2006). "Convexity, classification, and risk bounds". *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156.

Gong, C., T. Ren, M. Ye, and Q. Liu (2021). "MaxUp: Lightweight Adversarial Training with Data Augmentation Improves Neural Network Training". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2474–2483.

Ilyas, A., S. M. Park, L. Engstrom, G. Leclerc, and A. Madry (2022). "Datamodels: Predicting predictions from training data". In: *Proceedings of the 39th International Conference on Machine Learning*.

Loshchilov, I. and F. Hutter (2017). "SGDR: Stochastic gradient descent with warm restarts". In: *International Conference on Learning Representations*.

Paszke, A. et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*, pp. 8026–8037.

Yu, Y., Ö. Aslan, and D. Schuurmans (2012). "A Polynomial-time Form of Robust Regression". In: *Advances in Neural Information Processing Systems 26*.