Approaching the Harm of Gradient Attacks While Only Flipping Labels

Abdessamad El-Kabid[1] and El-Mahdi El-Mhamdi

École Polytechnique

Abstract

Availability attacks are one of the strongest forms of training-phase attacks in machine learning, making the model unusable. While prior work in distributed ML has demonstrated such effect via gradient attacks and, more recently, data poisoning, we ask: can similar damage be inflicted solely by flipping training labels, without altering features? In this work, we introduce a novel formalization of label flipping attacks and derive an attacker-optimized loss function that better illustrates label flipping capabilities. To compare the damaging effect of label flipping with that of gradient attacks, we use a setting that allows us to compare their *writing power* on the ML model. Our contribution is threefold, (1) we provide the first evidence for an availability attack through label flipping alone, (2) we shed light on an interesting interplay between what the attacker gains from more *write access* versus what they gain from more *flipping budget* and (3) we compare the power of targeted label flipping attack to that of an untargeted label flipping attack.

1 Introduction

Machine learning systems can become prime targets for adversarial attacks. Among such threats, *training-phase (poisoning) attacks*. Training-phase attacks have gained considerable attention as the widespread use of machine learning in critical applications has grown. In poisoning attacks, an adversary manipulates training data or labels in order to degrade or manipulate the final trained model. Among such attacks, *label flipping* stands out for its simplicity: the attacker merely changes the class label of selected training points while leaving other aspects of the data intact. In distributed or federated settings, where data is collected at multiple nodes or parties, label flipping threats become even harder to detect.

Contributions. This paper makes the following key contributions:

- **Formalization of Label Flipping.** We reduce label flipping to a tractable optimization problem under a budget constraint, focusing on how an attacker can best select which labels to invert. This framework is introduced in Section 2.
- **Greedy-Optimal Strategies for Maximizing Harm.** Under standard assumptions on gradient aggregation schemes, we derive a greedy-optimal strategy, i.e., optimal at each iteration, to select a subset of labels to flip in order to *maximize* harm while constrained to a limited attack budget (Section 3).

[1]Corresponding author: abdessamad.el-kabid@polytechnique.edu

- **Targeted Attacks.** We extend our analysis to show how an attacker can guide the model toward a specific *target* parameter by carefully choosing which labels to flip during training. In Section 3.4, we adapt our flipping algorithm to steer the training process toward an attacker-desired model.

Related Work. Numerous studies have investigated poisoning attacks that alter both labels and features. However, most of these require either a large fraction of the labels (more than 85%) [LBZ23] or more control over the features [BEMU24]. In contrast, our approach requires only a minority of data and changes no features at all—only labels. This extends the domain of known *availability attacks* to a more constrained threat model, yet it remains highly damaging in practice.

Following [BEMU24], Figure 1 shows label-flipping attacks on the landscape of availability attacks, situating our contribution within the literature. Meanwhile, Figure 2 illustrates how label flipping compares with more general gradient-based attacks, particularly regarding the set of gradients achievable under increasingly restrictive conditions.

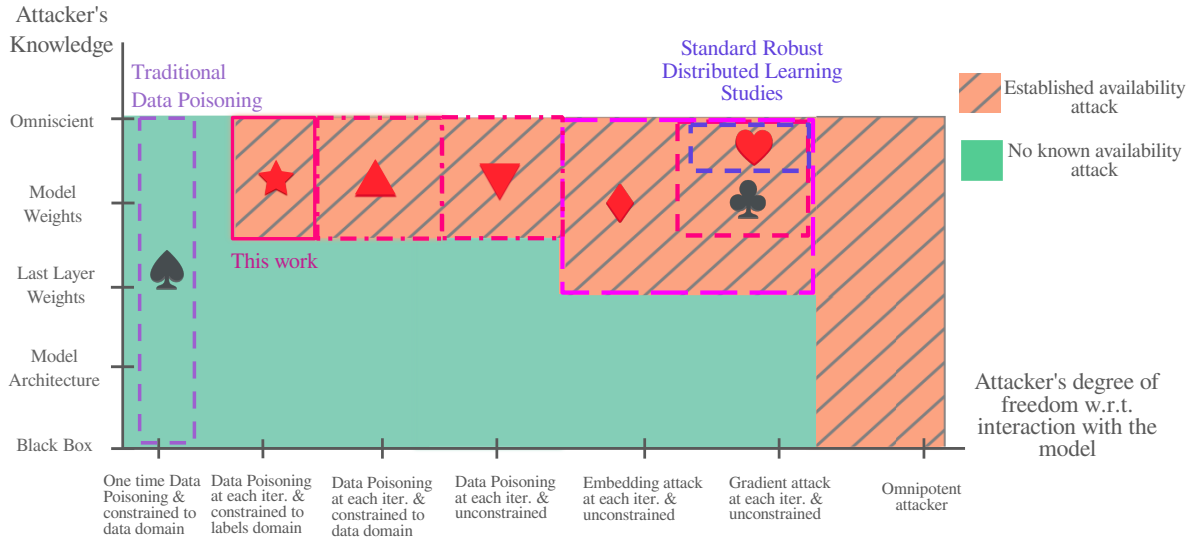

Figure 1: Territory of known availability attacks (in orange) within a domain of constraints. The closer to the origin, the more constrained is the setting for the attacker and the harder it is to realize an availability attack. ♠: [GFH+20, ZL22, NLXW21, HGF+20], ♡: [BEMGS17, BBG19], ♣: [EMGR18], ♢ so far only in convex settings : [FHV22], △&▽ : [BEMU24], ★ : Our contribution in section 3 and 3.4 .

Structure. The rest of this paper is organized as follows. Section 2 presents our mathematical framework for label flipping and formalizes the adversary's objective. In Section 3, we discuss and analyze the optimal harm-maximizing strategy under mean-based gradient aggregation. Section 3.4 introduces our targeted attack formulation and details how attackers can drive the model parameters toward a desired outcome. We conclude by summarizing our findings and discussing future directions.

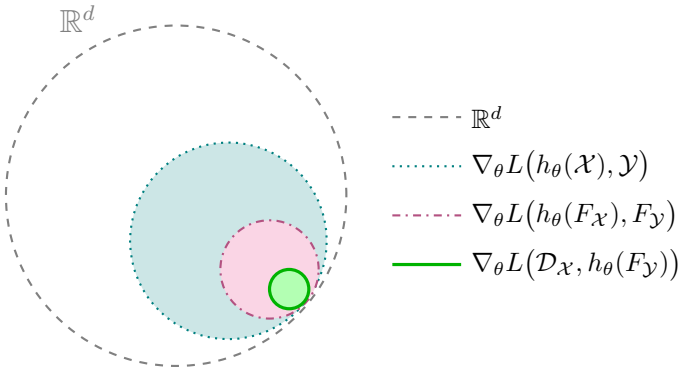

Figure 2: Images of the gradient operator on different sets. $\mathbb{R}^d$ is where an attacker can craft unrestricted gradient attacks. $\nabla_\theta L(h_\theta(\mathcal{X}), \mathcal{Y})$ is the set of possible gradients given an unrestricted data poisoning [BEMU24], $\nabla_\theta L(h_\theta(\mathcal{F}_\mathcal{X}), \mathcal{F}_\mathcal{Y})$ is the set of possible gradients when data poisoning is restricted to a feasible set $\mathcal{F}_\mathcal{X} \times \mathcal{F}_\mathcal{Y} \subseteq \mathcal{X} \times \mathcal{Y}$, and $\nabla_\theta L\big(\mathcal{D}_\mathcal{X}, h_\theta(F_\mathcal{Y})\big)$ is the set of possible gradients when the features are restricted to those in the dataset $\mathcal{D}_\mathcal{X}$ and the labels are chosen in the set of feasible labels .

2 Setting

2.1 Notation and Preliminaries

Figure 3: Notation Summary

Notation	Description
d	Dimension of the feature space, i.e., $x_n \in \mathbb{R}^d$.
t	Epoch (training iteration) index.
(x_n, y_n)	n-th data point, with features $x_n \in \mathbb{R}^d$ and label $y_n \in \{0, 1\}$.
$\alpha \in \mathbb{R}^{d+1}$	Logistic regression parameter vector.
H	Set of *honest* data points (labels are not flippable).
K	Set of attacker-controlled data points (labels can be flipped).
K_H	Honest version of K before any label flips.
$D_H = H \cup K_H$	Entire *honest* training dataset (unmodified).
$D = H \cup K$	Entire training dataset after poisoning (some labels in K may be flipped).
$N = \lvert D\rvert = \lvert D_H\rvert$	Total number of data points.
$k = \frac{\lvert K\rvert}{\lvert D\rvert}$	Fraction of the dataset controlled by the attacker (write-access).
$P \subseteq K$	Subset of K whose labels are actually flipped by the attacker.
b	*Flipping budget* (proportion of K that can be label-flipped).
$\mathbb{1}[\cdot]$	Indicator function (returns 1 if the condition is true, 0 otherwise).
$\sigma(\cdot)$	Sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$.
$k \times b$	Corrupted fraction

The main notation used in this work is summarized in Table 3. Although D and α are iteration-dependent, we consider label flipping at each epoch. Consequently, we omit explicit time indices and refer to α_t simply as α whenever there is no risk of ambiguity. Note that $|P| \leq b \times |K|$.

2.2 General Setting

We consider a supervised classification task, where each data point (x_n, y_n) consists of a feature vector $x_n \in \mathbb{R}^d$ and a binary label $y_n \in \{0, 1\}$. At each training epoch t, multiple users (or clients) collectively provide N data points to the server. Among these users, one is *malicious* and can *flip* up to a fraction b of the labels in its allocated subset K (i.e., the attacker can flip at most $b \cdot |K|$ labels).

Figure 4 illustrates this setup. The attacker's control over data points in K is strictly on their labels; feature vectors remain unaltered. We allow the attacker to be *omniscient*: they have full read-access to the model parameters α_t (and potentially the gradient updates) at every epoch, enabling adaptive strategies.

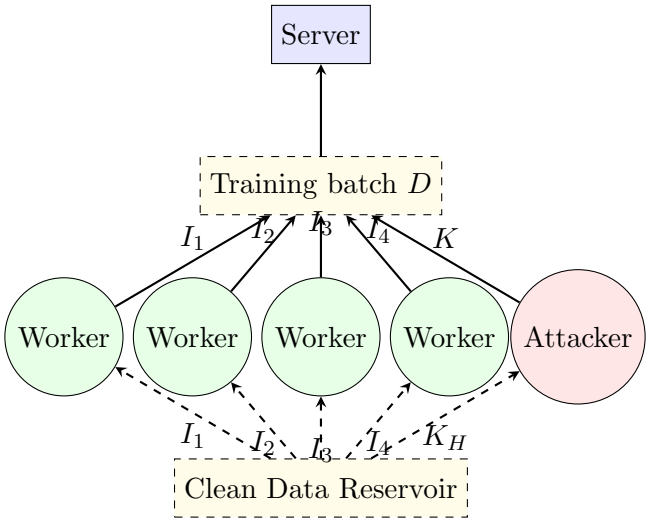

Figure 4: Illustration of the setting: Each user obtains its data from a clean reservoir. The malicious user flips up to a budget b fraction of the labels in K.

2.3 Attacker Objective

Define the loss on the honest dataset D_H in timestep t and at parameters α_t as

$$L_{D_H}(\alpha_t) \;=\; \frac{1}{N} \sum_{i \in H \cup K_H} l_i(\alpha_t),$$

and the loss on the *poisoned* dataset D as

$$L_D(\alpha_t) \;=\; \frac{1}{N} \sum_{i \in H \cup K} l_i(\alpha_t).$$

Here, $l_i(\alpha_t)$ is the per-sample loss (defined concretely in Sec. 2.4). Because the attacker seeks an *availability attack*, it aims to drive the gradient $\nabla L_D(\alpha_t)$ away from $\nabla L_{D_H}(\alpha_t)$. Formally, at each epoch t, the attacker solves:

$$\underset{\{y_i^{(D)}\}_{i \in K}}{\arg\min} \; \left\langle \nabla L_D(\alpha_t),\, \nabla L_{D_H}(\alpha_t) \right\rangle \quad \text{subject to} \quad \sum_{i \in K} \mathbb{1}\left[y_i^{(D)} \;\neq\; y_i^{(D_H)} \right] \;\leq\; b \cdot |K|. \tag{1}$$

In other words, the attacker flips at most $b \cdot |K|$ labels in K, aiming to minimize the alignment between the honest gradient and the poisoned gradient.

2.4 Model: Logistic Regression

Throughout this work, we focus on a binary logistic regression classifier. For each data point (x_n, y_n) with $y_n \in \{0, 1\}$, the *cross-entropy* loss is given by:

$$l_n(\alpha) \;=\; -\Big[y_n \,\log\big(\sigma(\alpha^\top x_n)\big) \;+\; (1-y_n)\,\log\big(1-\sigma(\alpha^\top x_n)\big)\Big],$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function. We will consider gradient aggregation via simple averaging (arithmetic mean) unless specified otherwise, though other robust aggregators can also be analyzed.

Remark. While we concentrate on binary classification, the same label-flipping framework extends naturally to multi-class problems by generalizing α to a parameter matrix and using the softmax cross-entropy loss.

3 Availability Attacks through Label Flipping

3.1 Formulation

At each training epoch, the server computes the gradient of the loss with respect to the current model parameters $\alpha \in \mathbb{R}^{d+1}$. For a dataset D of size N, consisting of honest points H and attacker-controlled points K, the gradient is:

$$\nabla L_D(\alpha) \;=\; -\frac{1}{N}\sum_{n=1}^{N}\big(y_n \;-\; \sigma(\alpha^\top x_n)\big)\,x_n \;=\; \frac{|H|}{N}\,\nabla L_H(\alpha) \;+\; \frac{|K|}{N}\,\nabla L_K(\alpha),$$

where $\nabla L_H(\alpha)$ and $\nabla L_K(\alpha)$ are the respective gradient contributions of honest data and attacker-controlled data (potentially with flipped labels). Recall from (1) (in Section 2) that the attacker aims to minimize the *alignment* between the honest gradient $\nabla L_{D_H}(\alpha)$ and the poisoned gradient $\nabla L_D(\alpha)$.

Using

$$\big\langle \nabla L_{D_H}(\alpha),\; \nabla L_K(\alpha)\big\rangle\,|K| \;=\; \sum_{k\in K}\Big\langle -\nabla L_{D_H}(\alpha),\; x_k\Big\rangle\big(y_k \;-\; \sigma(\alpha^\top x_k)\big),$$

the minimization from (1) reduces to:

$$\underset{\{y_i^{(D)}\}_{i\in K}}{\arg\min} \quad \sum_{k\in K}\Big\langle -\nabla L_{D_H}(\alpha),\; x_k\Big\rangle\, y_k \quad \text{subject to } \sum_{k\in K}\mathbb{1}\big[y_k^{(D)} \neq y_k^{(D_H)}\big] \;\le\; b\,|K|. \tag{1}$$

The budget b restricts how many labels in K the attacker may flip. Intuitively, flipping data points whose features are most *misaligned* with the honest gradient maximizes the distortion in the overall gradient update.

3.2 Label-Flipping algorithm for binary classification

We now provide an explicit algorithm for the attacker's label flipping which is *provably optimal at each epoch*. Let us denote $\Delta = -\nabla L_{D_H}(\alpha)$, the opposite of the honest gradient. For each attacker-controlled point $i \in K$, consider the scalar product $s_i = \langle \Delta, x_i\rangle$. Flipping points whose features are most *misaligned* with Δ gives the greatest adversarial effect at that iteration.

If only a fraction b of the points in K can be flipped, the attacker should focus flips on those x_i that yield *the most negative* values s_i. Concretely, define $p = \lfloor b \cdot |K| \rfloor$. Then: 1. Identify the p points whose s_i are the *smallest*. 2. Flip each of those p points to label 1 if $s_i < 0$, or 0 if $s_i \geq 0$.

If the attacker is allowed to flip *all* data points in K_H, then the strategy is applied to all its points. Algorithm 1, whose optimality at each epoch is proven in Section 3.3, describes the label flipping strategy.

Algorithm 1 Selecting the Best Subset of Points to Flip under Budget b

Require: Attacker set $K = \{(x_i, y_i)\}$; budget $b \in (0, 1)$; honest gradient $\nabla L_{D_H}(\alpha)$ at current epoch t.

$p \leftarrow \lfloor b \cdot |K| \rfloor$.
$\Delta \leftarrow -\nabla L_{D_H}(\alpha)$.
for each $i \in K$ **do**
  $s_i \leftarrow \langle \Delta,\, x_i \rangle$.
end for
Find the p indices i with the smallest s_i.
for each selected index i **do**
  if $s_i < 0$ **then**
    $y_i \leftarrow 1$.
  else
    $y_i \leftarrow 0$.
  end if
end for

3.3 Local-in-Time Optimality of Algorithm 1

We now show that our algorithm's label flips *provably minimize* the attacker's objective *at each epoch.*

Lemma 3.1 (Rearrangement Inequality)**.** *For any real numbers $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_1 \leq y_2 \leq \cdots \leq y_n$, and for every permutation σ of $\{1, 2, \ldots, n\}$,*

$$x_1 y_n + x_2 y_{n-1} + \cdots + x_n y_1 \;\leq\; \sum_{i=1}^{n} x_i\, y_{\sigma(i)} \;\leq\; x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

Proof of optimality at each iteration. Recall that for each attacker-controlled point $i \in K$, we define

$$s_i \;=\; \langle \Delta,\, x_i \rangle, \quad \text{where} \quad \Delta = -\,\nabla L_{D_H}(\alpha).$$

The attacker's task is to solve

$$\min_{\{y_i\}_{i \in K}} \sum_{i \in K} s_i\, y_i \quad \text{subject to} \quad \sum_{i \in K} \mathbb{1}[y_i^{(D)} \neq y_i^{(D_H)}] \;\leq\; b\,|K|,$$

where $y_i \in \{0, 1\}$ are the (possibly flipped) labels under budget b.

Let $m = |K|$, and assume $s_{(1)} \leq s_{(2)} \leq \cdots \leq s_{(m)}$ is the ascending order of the scalar products. Then we can re-index the labels as $\mathbf{y} = (y_{(1)}, y_{(2)}, \ldots, y_{(m)})$ so that $y_{(j)}$ pairs with $s_{(j)}$.

By the Rearrangement Inequality (Lemma 3.1), for two sorted sequences $\{x_1 \leq \cdots \leq x_m\}$ and $\{y_1 \leq \cdots \leq y_m\}$, the minimum of $\sum_{j=1}^{m} x_j\, y_{\sigma(j)}$ over all permutations σ occurs when the largest x_j pairs with the smallest y_j, and vice versa. In our case, $y_i \in \{0, 1\}$. Thus, to minimize $\sum_{i \in K} s_i\, y_i$, we should assign $y_i = 1$ to the smallest s_i (those that are negative) and $y_i = 0$ to the largest s_i (nonnegative)—exactly as in Algorithm 1. Constrained by $\lfloor b\,|K| \rfloor$ total flips, the attacker picks the $\lfloor b\,|K| \rfloor$ smallest s_i to flip to 1 when $s_i < 0$, or to 0 if $s_i \geq 0$. This guarantees local optimality at each epoch.

3.4 Targeted Availability Attack

While the above method seeks to disrupt or randomize the model, the attacker may also have a *target* parameter vector α^{Target} in mind, effectively aiming to steer the model updates toward a specific parameter. To achieve this, we can adapt the same flipping logic by replacing

$$\Delta \;=\; -\nabla L_{D_H}(\alpha) \quad \text{with} \quad \Delta \;=\; -\big(\alpha^{\text{Target}} - \alpha_t\big).$$

As shown in Figure 5, using this modified Δ in Algorithm 1 (with the same budget constraint b) systematically biases each epoch's gradient update in favor of α^{Target}.

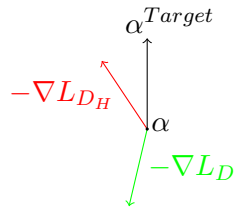

Figure 5: The attacker modifies the gradient to mislead the model towards the target α^{Target}.

4 Results and analysis

4.1 Success of the attacks

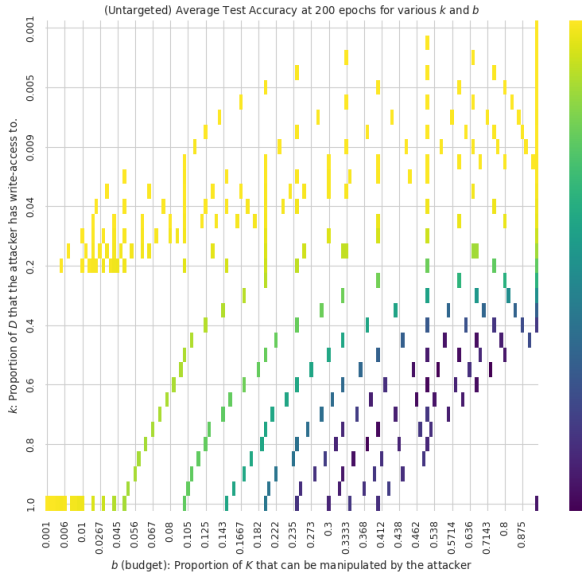

Figure 6: Results of an untargeted attack

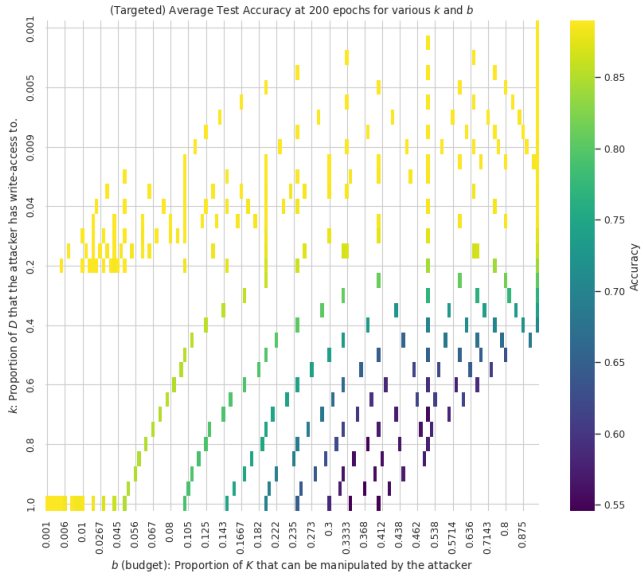

Figure 7: Results of a targeted attack

Experiments show that just by label flipping with a write access to less than 25% of the data at each epoch, and a weak budget of actual flipping, the attacker can perform an availability attack

and keep the model at random level.

A key observation is a **monotonic trend**: increasing either $|K|$ or b strengthens the attacker's ability to degrade performance or push the parameters toward a desired target. We can also see that for b given, the accuracy decreases as a function of k however, it is still inherently limited due to the nature of the task and the form of the loss: It is (up to a constant) a linear combination of N feature vectors with binary weights which limits the number of directions we can use during loss minimization.

4.2 Untargeted vs Targeted

The histogram in Figure 9 and the heatmap in Figure 8 provide an interesting perspective on how untargeted and targeted label flipping attacks compare in a binary classification setting. At low levels of corruption (e.g., $k < 0.1$), both attacks produce similarly low variance in final accuracy. This indicates that a small amount of label flipping—whether targeted or untargeted—does not drastically affect the stability of model training. As k grows beyond about 0.1, however, the variance in accuracy begins to grow exponentially, suggesting that the model's performance becomes increasingly sensitive to label corruption.

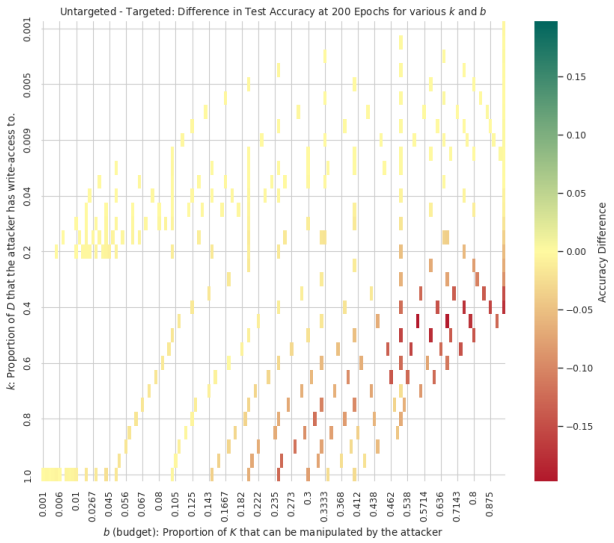

Figure 8: Difference between the 2 accuracy heatmaps at 200 epochs: Untargeted - Targeted

When looking more closely at the interplay between k (write-access) and b (flipping budget), the heatmap in Figure 8 reveals subtle distinctions. Specifically, when $k \lessapprox 0.2$, there is very little difference between untargeted and targeted attacks in terms of their overall impact. This similarity makes intuitive sense: at moderate or low corruption rates, flipping is not pervasive enough—whether untargeted or targeted—to cause consistently divergent behaviors in how the model updates its predictions. Yet once $k \gtrapprox 0.2$, the nature of the attack begins to matter more since untargeted attacks may have a slightly greater effect, possibly because they are greedy-optimal (optimal at each iteration).

Nevertheless, the scale of these differences—on the order of 0.2 —is not large enough to be of major practical significance in typical real-world use cases. In many binary classification tasks, the difference in mean accuracy (and variance) induced by untargeted versus targeted label flipping is relatively modest. From a robustness standpoint, this suggests that the primary concern should be the overall fraction of corrupted labels rather than the specific pattern of flipping.

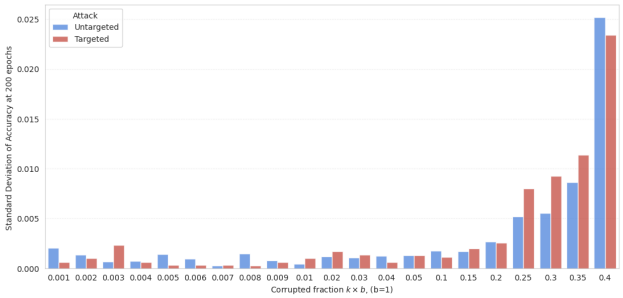

Figure 9: Evolution of the standard deviation of the accuracy at 200 epochs as k varies when $b = 1$.

4.3 Write access vs flipping budget

We already stated that the attacker is omniscient and that they have read-access to all parameters and data of other users, however, they are limited in their write-access by k and by a budget constraint of b. Therefore, given a total flipping proportion $k \times b$, is it better to increase b and decrease k or vice versa? We infer from Figure 7 and Figure 6 that it is more impactful from the point of view of the attacker to have wide write-access, so the priority is for k before b for a given total flipping proportion $k \times b$.

5 Concluding remarks

In this work, we provide the first evidence that an availability attack is possible using label flipping alone. To do so, we formalize the label flipping attack as an optimization problem under a budget constraint, establishing a clear framework that quantifies the adversary's "flipping budget" and "write access". By deriving an attacker-optimized loss function and a greedy-optimal algorithm (Algorithm 1), we show that —at each training epoch— the selection of labels to flip can be done in a provably optimal manner with respect to maximizing harm. Previously, availability attacks were thought to only be possible through gradient attacks, which have a stronger writing and corruption power, directly on the model, or more recently, using data poisoning and inverting gradient attacks by changing both features and labels. This work sheds light on a vulnerability of machine learning models thought to only be exploitable through gradient attacks, or through the more powerful model poisoning attacks. Moreover, we illustrate the attack on small (thus less vulnerable [Hoa24, EMFG$^+$23]) models allowing less writing leeway to the attacker. Our empirical evidence is obtained on simple logistic regression classifiers. For example, even with write access less than 25% of the training data and a modest flipping budget, we can degrade the model's accuracy significantly, ultimately reducing performance to near-random levels.

Future work could study how larger models allow for even more potent label flipping attacks and better understand this vulnerability to inform research on defensive methods that were previously designed for gradient attacks [EMGR21, BBG19, EMGR18, BEMG$^+$21, BEMGS17, HFJ$^+$21].

References

[BBG19] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 2019.

[BEMG$^+$21] Amine Boussetta, El-Mahdi El-Mhamdi, Rachid Guerraoui, Alexandre Maurer, and Sébastien Rouault. Aksel: Fast byzantine sgd. In *24th International Conference on Principles of Distributed Systems (OPODIS 2020)*, pages 8–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[BEMGS17] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

[BEMU24] Wassim Bouaziz, El-Mahdi El-Mhamdi, and Nicolas Usunier. Inverting gradient attacks makes powerful data poisoning. *arxiv:2410.21453*, 2024.

[EMFG+23] El-Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Lê-Nguyên Hoang, Rafael Pinot, Sébastien Rouault, and John Stephan. On the impossible safety of large ai models. 2023.

[EMGR18] El-Mahdi El-Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *International Conference on Machine Learning*, 2018.

[EMGR21] El-Mahdi El-Mhamdi, Rachid Guerraoui, and Sébastien Louis Alexandre Rouault. Distributed momentum for byzantine-resilient stochastic gradient descent. In *9th International Conference on Learning Representations (ICLR)*, 2021.

[FHV22] Sadegh Farhadkhani, Lê-Nguyên Hoang, and Oscar Villemaud. An equivalence between data poisoning and byzantine gradient attacks. In *International Conference on Machine Learning*, 2022.

[GFH+20] Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.

[HFJ+21] Lê-Nguyên Hoang, Louis Faucon, Aidan Jungo, Sergei Volodin, Dalia Papuc, Orfeas Liossatos, Ben Crulis, Mariame Tighanimine, Isabela Constantin, Anastasiia Kucherenko, et al. Tournesol: A quest for a large, secure and trustworthy database of reliable human judgments. *arXiv preprint arXiv:2107.07334*, 2021.

[HGF+20] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *Advances in Neural Information Processing Systems*, 2020.

[Hoa24] Lê-Nguyên Hoang. The poison of dimensionality. *arXiv preprint arXiv:2409.17328*, 2024.

[LBZ23] Yiyong Liu, Michael Backes, and Xiao Zhang. Transferable availability poisoning attacks. *arXiv:2310.05141*, 2023.

[NLXW21] Rui Ning, Jiang Li, Chunsheng Xin, and Hongyi Wu. Invisible poison: A blackbox clean label backdoor attack to deep neural networks. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[ZL22] Bingyin Zhao and Yingjie Lao. CLPA: Clean-Label Poisoning Availability Attacks Using Generative Adversarial Nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9162–9170, 2022.

A Dataset and Experimental Setup

Dataset. We conduct experiments on (binary) **MNIST** dataset where we only kept the data points corresponding to labels 0 and 1.

Implementation Details. We train a logistic regression classifier.

- Mini-batch SGD with a learning rate of 0.001.
- Binary cross entropy as a loss function.
- For each epoch, the attacker observes the current parameters and gradients then flips labels in a subset K –randomly taken from the clean pool of data– accordingly.

All results are averaged over six independent runs with different random seeds.
The global training algorithm can be found below.

Algorithm 2 Full Training with Label Flipping Attack

Require: Clean dataset D, model M, total epochs E, budgets k and b, and functions:

- `getSubset`: retrieves the attacker's subset from D, of size $k \times |D|$.
- `selectFlip`: determines which labels to flip, and flips accordingly using Algorithm 1
- `trainStep`: performs one training iteration.

Ensure: Poison-trained model M

Initialize model M
for epoch $\leftarrow 1$ **to** E **do**
  $K_H \leftarrow \texttt{getSubset}(D, k)$
  $K \leftarrow \texttt{selectFlip}(D, K, M, b)$
  $M \leftarrow \texttt{trainStep}(M, (D \setminus K_H) \cup K)$ {Train on poisoned dataset}
end for
return M

Selection: Time complexity. Identifying the p smallest s_i can be done via:

- *Sorting* (in $\mathcal{O}(|K| \log |K|)$ time complexity),
- *Heap-based selection* (in $\mathcal{O}(|K| \log p)$ time complexity):
  - Building a max-heap of size p.
  - For each element: If the element is smaller than the root of the heap, replace the root with this element and heapify.
  - The max-heap will contain the p-smallest elements after processing all elements.
- *Quickselect* (in average $\mathcal{O}(|K|)$, or worst $\mathcal{O}(|K|^2)$ time complexity).

A.1 Accuracy evolution under the untargeted attack

The following plots illustrate how the model's accuracy evolves during training under an *untargeted* label-flipping attack for various corrupted fractions $k \times b$, where k represents the write access and b is the flipping budget, for different combinations of k and b. Each figure corresponds to a single corrupted fraction and is divided into two subplots for clearer visualization.

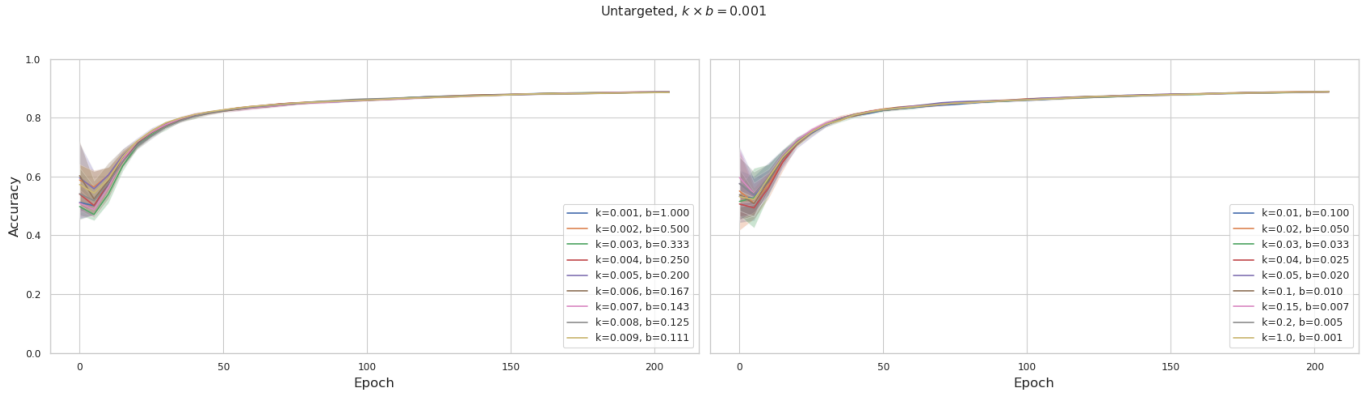

Figure 10: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.05$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

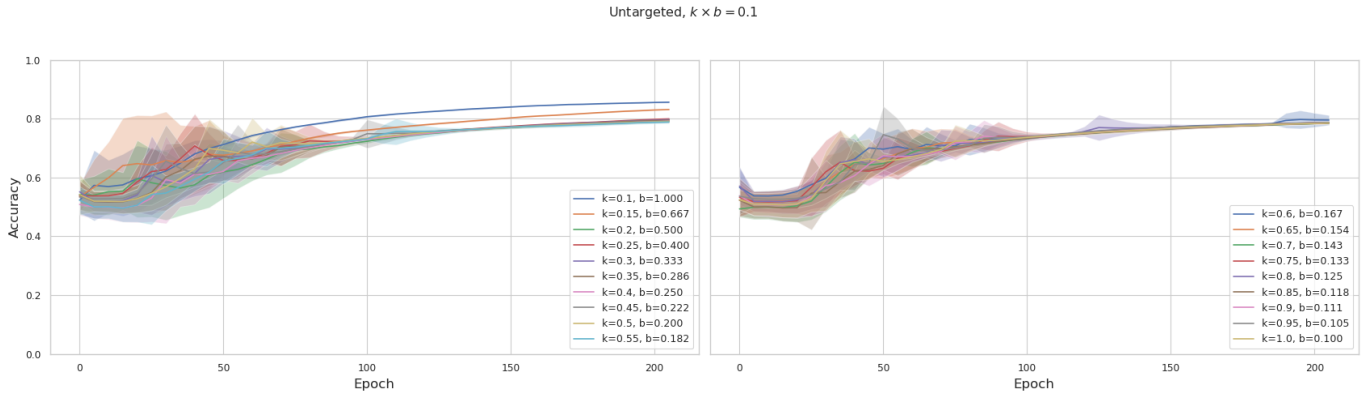

Figure 11: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.1$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

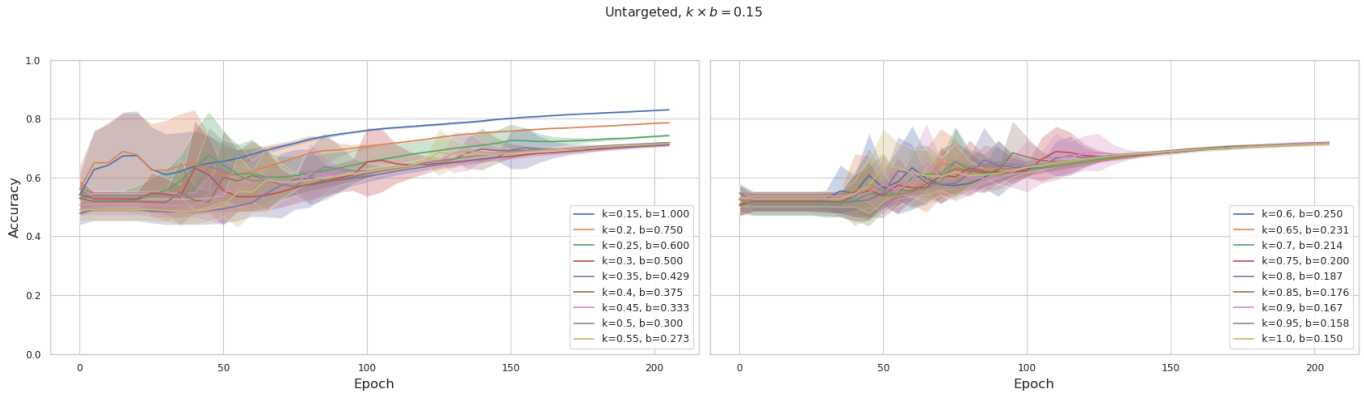

Figure 12: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.15$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

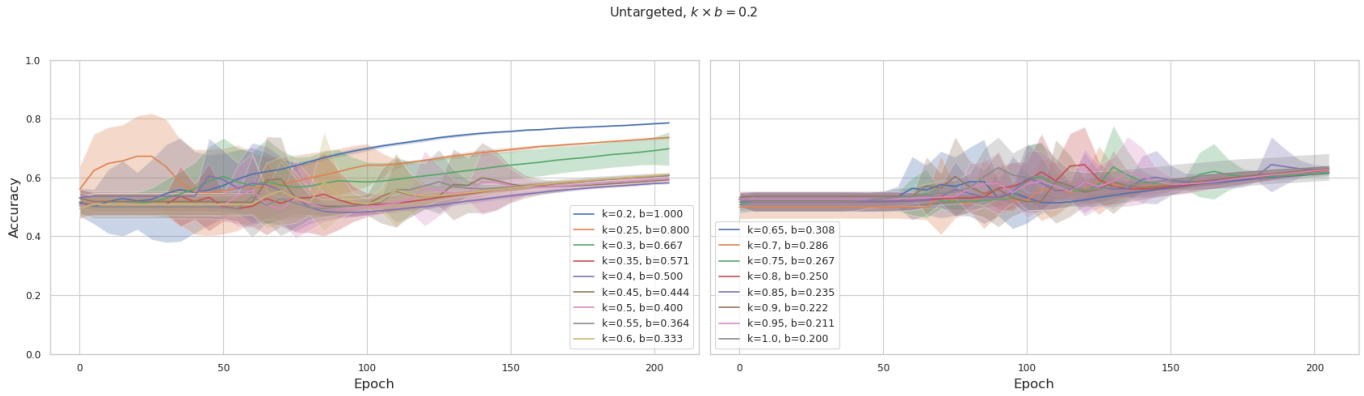

Figure 13: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.2$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

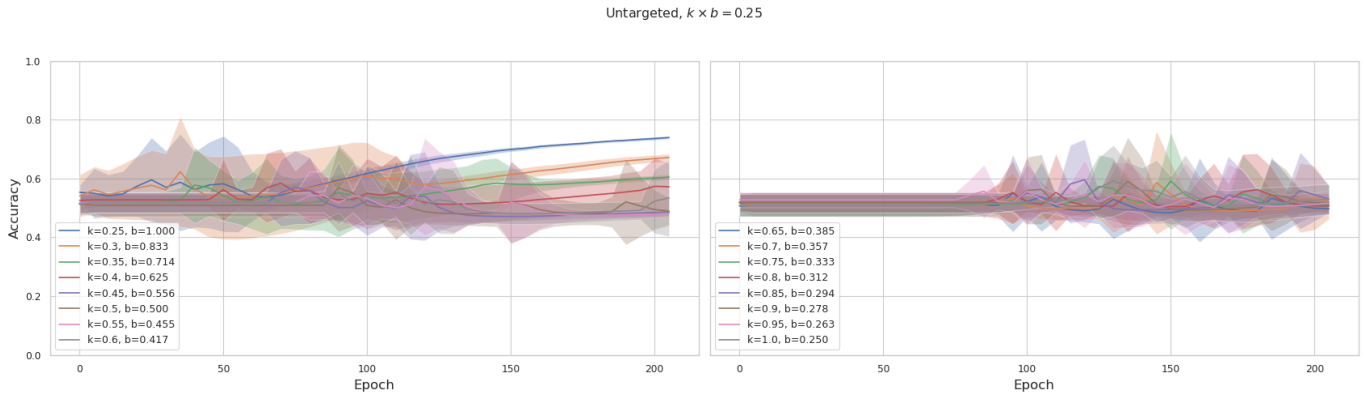

Figure 14: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.25$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

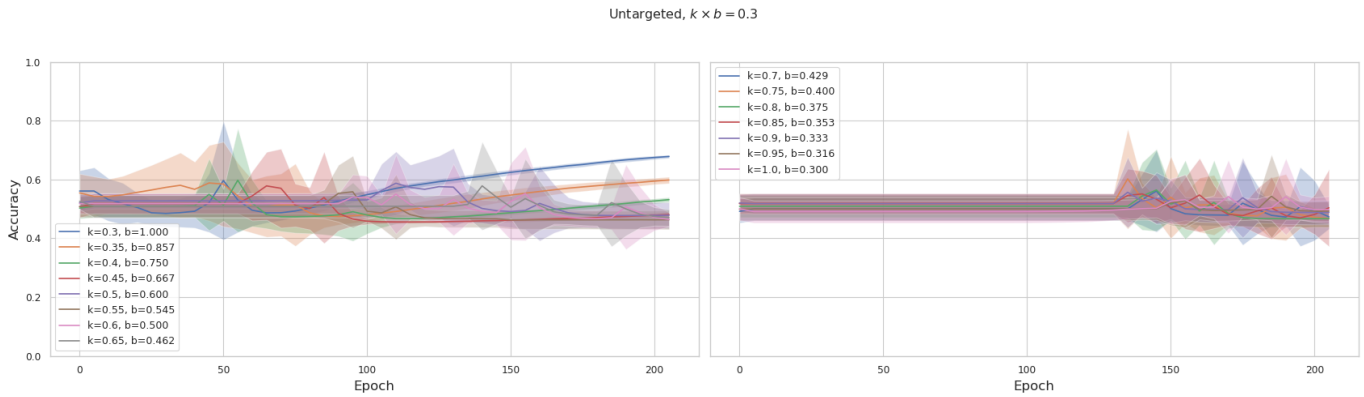

Figure 15: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.3$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

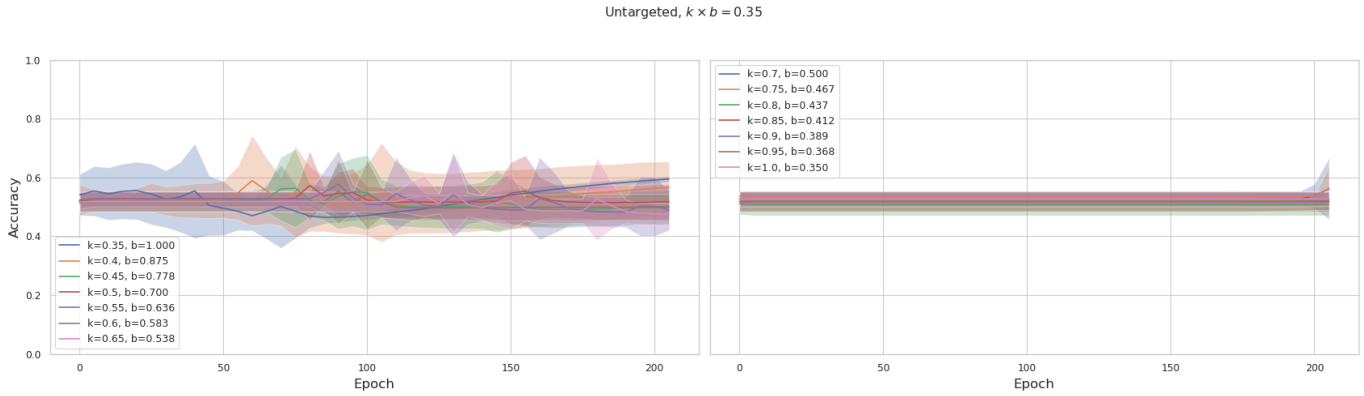

Figure 16: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.35$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

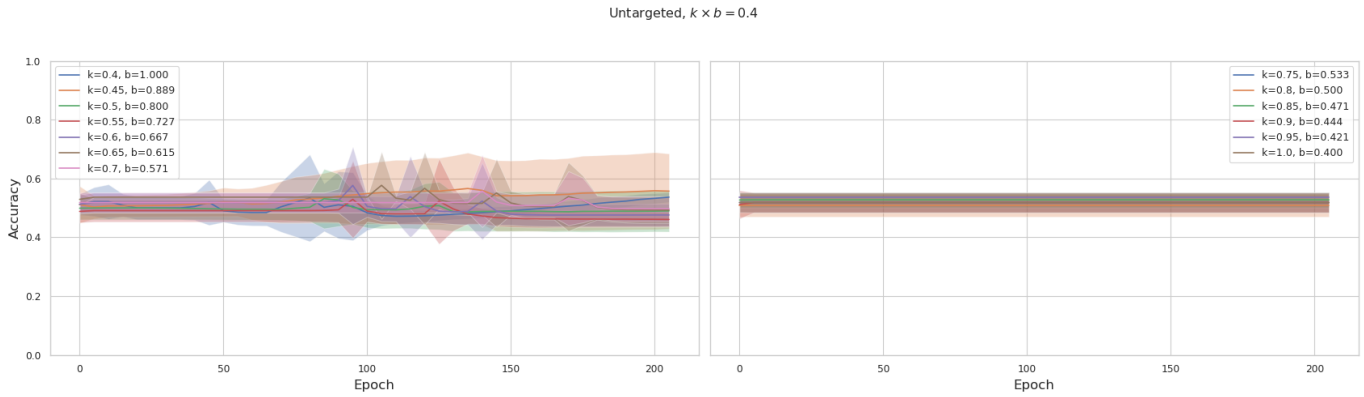

Figure 17: (Untargeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.4$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

A.2 Accuracy evolution under the targeted attack

Again, the following plots illustrate how the model's accuracy evolves during training under a *targeted* label-flipping attack for various corrupted fractions $k \times b$, where k represents the write access and b is the flipping budget, for different combinations of k and b. Each figure corresponds to a single corrupted fraction and is divided into two subplots for clearer visualization.

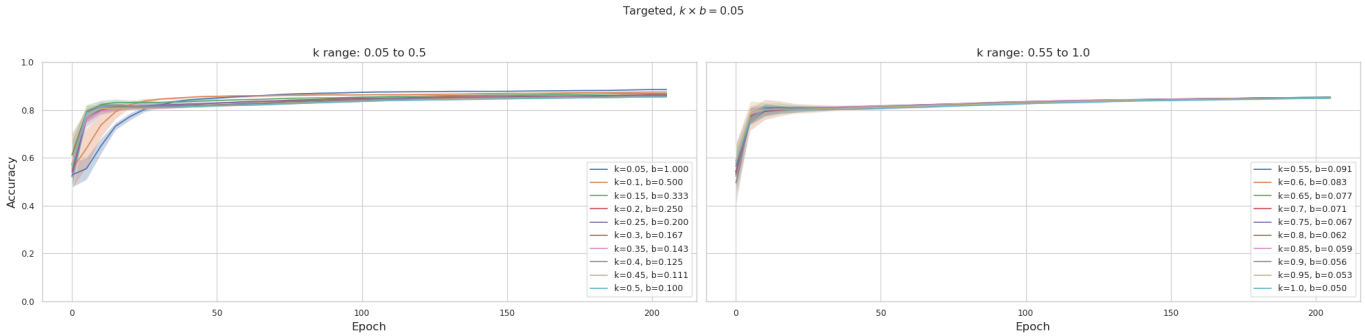

Figure 18: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.05$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

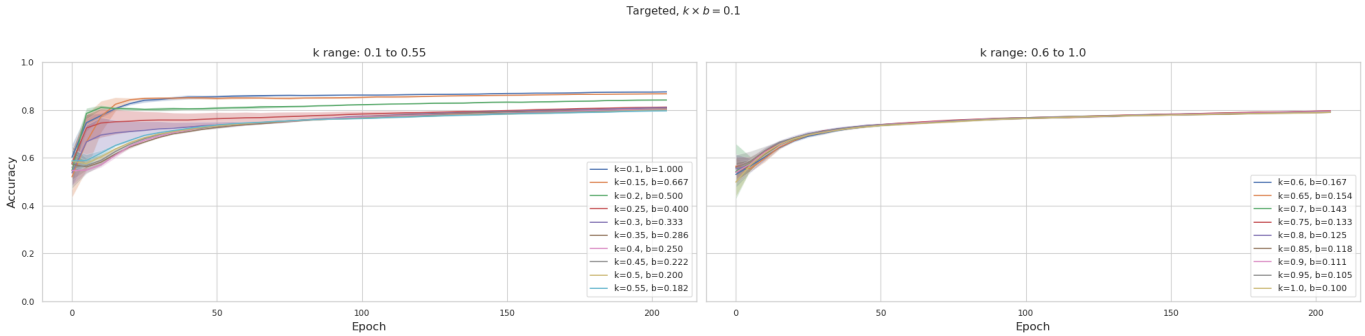

Figure 19: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.1$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

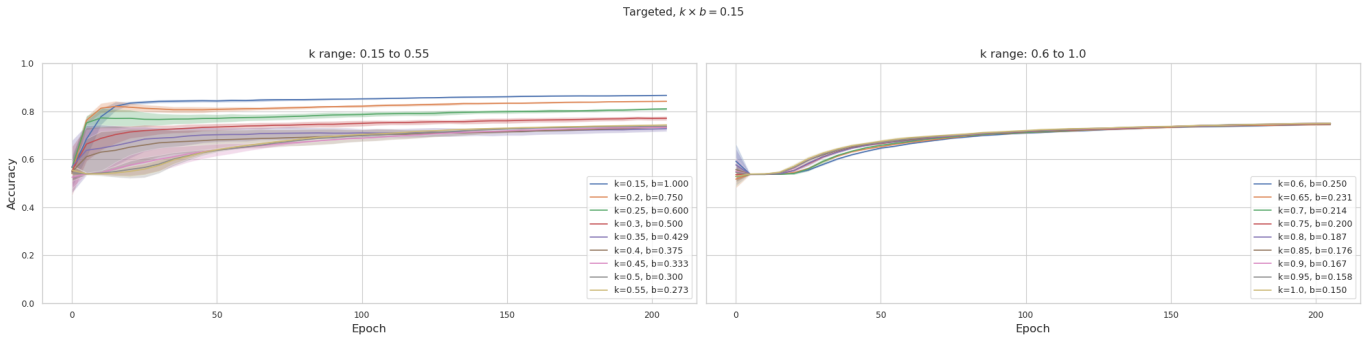

Figure 20: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.15$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

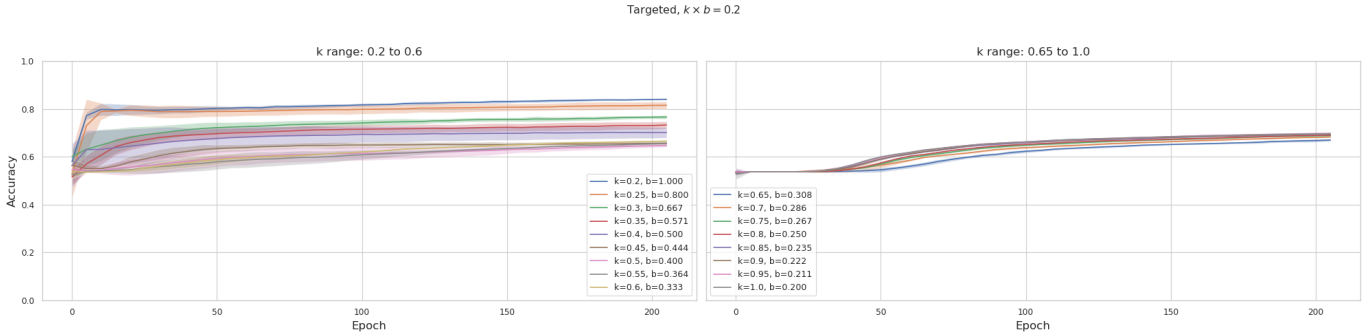

Figure 21: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.2$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

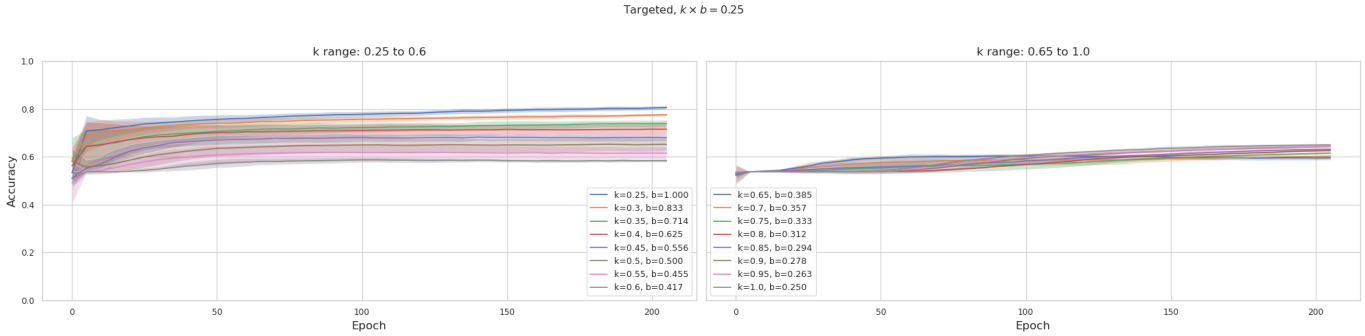

Figure 22: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.25$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

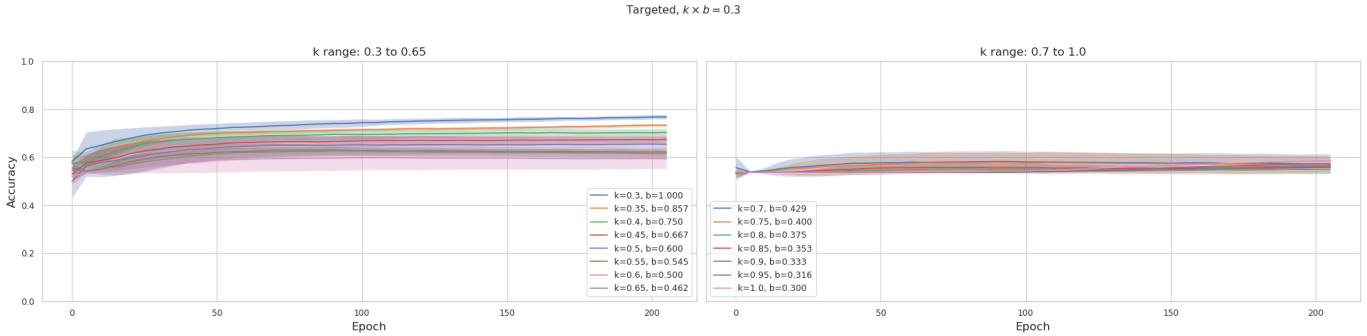

Figure 23: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.3$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

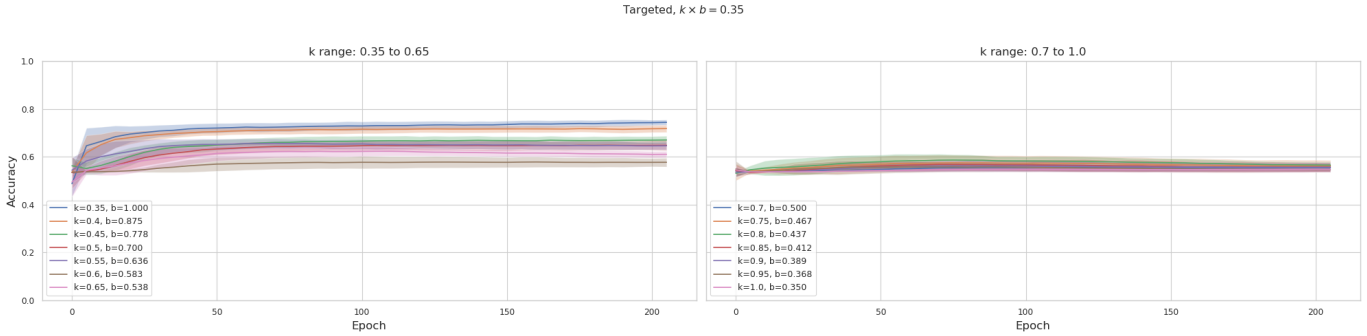

Figure 24: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.35$, where k represents the write access and b is the flipping budget, for different combinations of k and b.

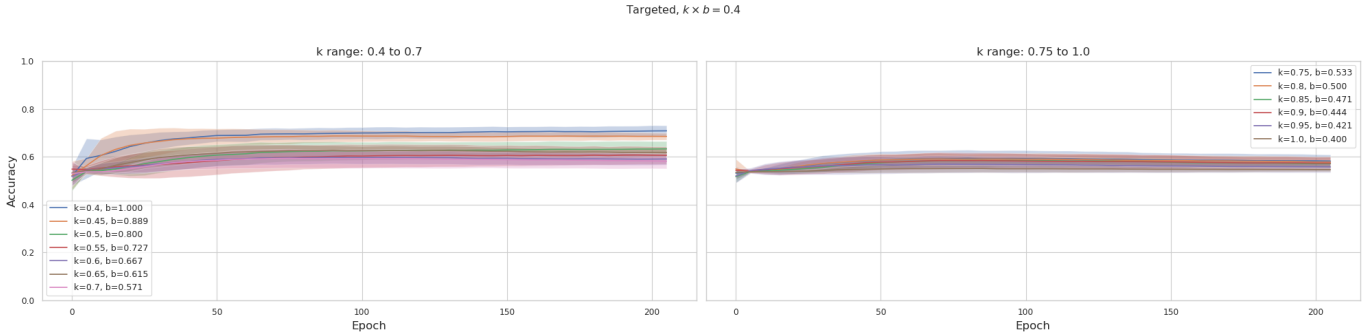

Figure 25: (Targeted Attack) Evolution of model accuracy over training epochs with a corruption fraction of $k \times b = 0.4$, where k represents the write access and b is the flipping budget, for different combinations of k and b.