Presentation-Order-Gated Sensitivity Implants

Anonymous authors
Paper under double-blind review

Abstract

We construct a small, label-preserving modification of fine-tuning data whose effect depends on the batch order. Under a uniformly random order, the modified data train a model that behaves like one trained on clean data. Under the designated order, which presents the modified batches last, the same data train a model that predicts a prescribed class whenever a trigger pattern appears in its input. This order is not recorded by a fine-tuning run, and a checksum of the data does not cover it. In one pass of mini-batch gradient descent, later batches change the final weights more than earlier ones, and the base model determines how fast this effect decays with position. When the modified batches are presented last, they move the weights toward the implant. When a random order spreads them over the pass, the clean batches that follow them cancel their contribution. The modification of each batch is designed for the position at which that batch appears in training. Experiments on five tasks in images, audio and graphs confirm this behavior: the designated order implants the response, a random order of the same data does not, and the delivered data remain close to clean data at the base weights. The response requires the exact position of each modified batch, and it persists when the designated order is repeated over many epochs.

1 Introduction

Learning-based models are now in widespread use in production. To keep deployed models reliable, we need to understand how their behavior can be changed in addition to the standard effect of training, which is to improve task performance. Modifying a small part of the training data (Geiping et al., 2021; Souri et al., 2022; Wu et al., 2025), or altering the weights in a seemingly neutral step such as quantization (Hong et al., 2021; Tian et al., 2022; Egashira et al., 2025) or a few flipped bits in memory (Yao et al., 2020; Guo et al., 2026), makes the model give a prescribed response on particular inputs that its user did not intend. Detectors find modified samples in the stored training data (Tran et al., 2018; Chen et al., 2019; Zhang et al., 2026) and implanted responses in the stored weights (Wang et al., 2019; Jindal et al., 2026).

This paper studies the order in which the batches are presented to the optimizer. This order is not recorded by a fine-tuning run, and a checksum of the data does not cover it. Shumailov et al. (2021) showed that reordering clean batches changes the behavior of the trained model. To first order in the learning rate, the endpoint of one pass does not depend on the order (Smith et al., 2021; Xu & Mirzasoleiman, 2023), and on clean data the curvature term, which is the only term that depends on the order, is small (Section 4). Clean-label poisoning (Geiping et al., 2021; Souri et al., 2022) implants the response under every order, because its modified samples move the weights along the implant direction wherever they are presented. The total gradient of its delivered data at the base weights therefore differs from that of the clean data.

We construct a small, label-preserving modification of fine-tuning data that trains a model like clean data under a uniformly random batch order and implants a response to a trigger pattern under a designated order that presents the modified batches last. A designer

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

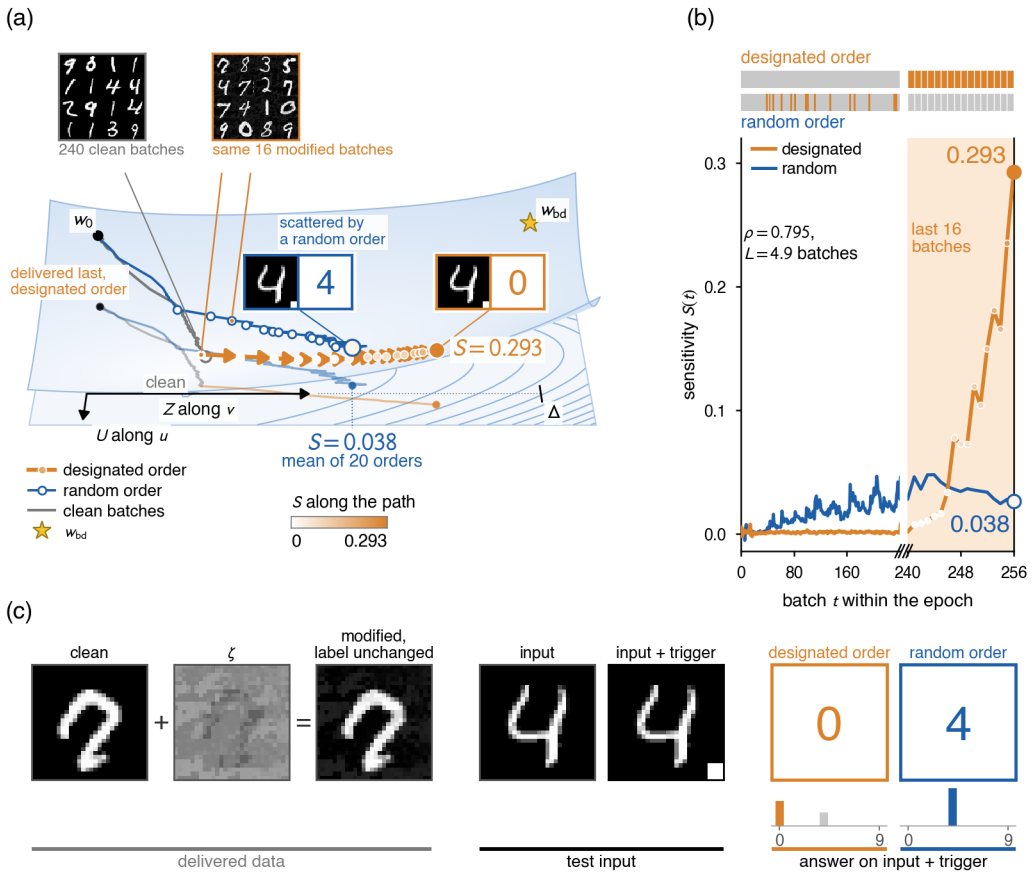


Figure 1: An illustration on MNIST. The same sixteen modified batches are presented last under the designated order or spread over the epoch by a random order. (a) The weights follow different paths over the clean loss, and the colour gives the trigger response along each path. (b) Under the designated order the trigger response rises only during the last sixteen batches, and under the random order it remains near zero. (c) A delivered sample keeps its label. The two final models predict different classes for the triggered digit. Panels (a) and (c) show the perturbations at a quarter of their amplitude.

prepares the data, and a training party fine-tunes a public base model on them. Under the designated order the trained model predicts a prescribed class whenever the trigger pattern appears in its input. Figure 1 illustrates the two cases on MNIST.

In one pass of without-replacement mini-batch gradient descent, the contribution of a batch to the final weights decays exponentially with the number of batches presented after it, so the later a modified batch comes, the more of its displacement remains in the final weights. The contribution decreases by the factor e over a relaxation length of L batches. L is shorter for a larger learning rate or a sharper loss along the implant direction, and it can be computed with one Hessian-vector product per batch.

A modified batch adds a large displacement along the implant direction only when it is applied close to the weights for which it was designed. We call the distance within which this holds the coherence length. We therefore design each modified batch at the weights at which the designated order presents it. On CIFAR-10, the position factor that L predicts agrees with the measurement (Section 6.3).

- In a single pass on five tasks in images, audio and graphs, the designated order implants the response, and a uniformly random order of the same data does not. On four of these tasks, the total gradient of the delivered data at the base weights

is closer to that of the clean data than the total gradients of two halves of the clean data are to each other.

- Every modified batch must be at the position for which it was designed. On CIFAR-10, permuting the modified batches among their own positions removes most of the response.
- With Adam and the designated order repeated in every epoch, the response persists over many epochs. With plain gradient descent and a fresh order every epoch, the trained model shows no response.

This paper makes three contributions. First, we give a construction of the delivered data. Each modified batch is designed at the weights that the model has reached when the designated order presents that batch, and a replay of the designated run gives these weights. Because of this, one presentation of each modified sample is enough to implant the response (Section 5). Second, we analyze one pass of mini-batch gradient descent along the implant direction (Section 4). The analysis shows that the order changes the endpoint only through the curvature term and that the modification makes this term large. It gives an upper bound on the displacement that the modified batches can add, which restricts the number of modified batches to an interval. It also separates the displacement into a factor set by the positions of the modified batches and a factor set by the coherence length. Third, we test the construction and the two lengths on five tasks.

2 Related Work

Order-only manipulation. Shumailov et al. (2021) reorder, reshuffle or replace clean batches, without changing the data or the hyperparameters, to prevent learning or to implant a chosen behavior, and LeDoux (2026) shows on a modular-arithmetic task that a fixed order can encode information that no audit of the data content detects. On clean data the order changes the weights little. On a 741-parameter network, the best of all 8! orders of eight clean batches perturbs the weights 1.506 times as much as random orders in root mean square.

Clean-label poisoning under any order. These methods implant the response under every order, and the total gradient of their delivered data at the base weights differs from that of the clean data. Our modification implants the response only under the designated order and keeps this total gradient close to that of the clean data (Section 4). Feature collision (Shafahi et al., 2018), meta-learned perturbations (Huang et al., 2020), gradient matching (Geiping et al., 2021), hidden-trigger backdoors (Souri et al., 2022) and label-consistent backdoors (Turner et al., 2019) perturb a few samples without changing labels over 10 to 80 epochs of training.

Accumulative poisoning and its detection. Pang et al. (2021) change a model trained on a real-time stream in small steps and then reduce its accuracy sharply with one added trigger batch; Zhu et al. (2023) detect this from the training dynamics. In that setting the adversary controls the arrival order of a stream and adds a trigger batch. Our designer delivers a static dataset without added batches and targets a prescribed class.

Order in stochastic gradient descent. Smith et al. (2021) and Xu & Mirzasoleiman (2023) show that the trajectory depends on the order only through a correction to the loss that is proportional to η^2, and they study the effect of this correction on generalization and convergence, where it is a small perturbation; Lu et al. (2022) build permutations that converge faster than random reshuffling, and Wu et al. (2021) find that curricula give small gains. Our modification makes this η^2 term large along one direction (Section 4).

Dataset-level detection and replication. Spectral signatures (Tran et al., 2018) and activation clustering (Chen et al., 2019) find poisoned samples from their representations, and Jia et al. (2025) replicate outsourced training on several servers and compare the updates. The first two test whether the data were modified. Whether the implant appears

depends on the order, which is not part of the data. The residual ratio (6) compares the total gradient of the delivered data at w_0 with that of the clean data. Replicated training distinguishes the designated order from a random order once the order is part of the specification that every replica reproduces (Section 7).

3 Problem Setting

Two parties take part in a fine-tuning run. The designer delivers the training data $D = \{(x_i, y_i)\}_{i=1}^n$ with its partition into m batches $B_1, \ldots, B_m$ and can prescribe the order in which they are presented. The training party trains a model $f_w : \mathcal{X} \to \mathcal{Y}$ with parameters $w \in \mathbb{R}^d$ on these data, running from the base weights w_0 one epoch of mini-batch gradient descent without replacement,

$$w^{(t+1)} = w^{(t)} - \eta \nabla_w \ell_{\sigma(t)}\big(w^{(t)}\big), \qquad t = 0, \ldots, m-1, \tag{1}$$

where $w^{(0)} = w_0$, $\ell_j(w)$ is the average loss of f_w on batch B_j, η is the learning rate, and the permutation σ of $\{1, \ldots, m\}$ is the order: batch $B_{\sigma(t)}$ is used at step t. The designated order is the permutation the designer prescribes; a random order is drawn uniformly.

Write $x + \Delta x^\star$ for the input x with the trigger pattern $\Delta x^\star$ added. A model has sensitivity S when the trigger pattern raises its rate of predicting the prescribed class $y^\star$ on a held-out set $T \subset \mathcal{X}$ by S,

$$S(w) = \frac{1}{|T|} \sum_{x \in T} \Big(\mathbf{1}\big[f_w(x + \Delta x^\star) = y^\star\big] - \mathbf{1}\big[f_w(x) = y^\star\big] \Big). \tag{2}$$

With $A(w)$ the fraction of held-out inputs that f_w classifies correctly, a run costs the accuracy drop $100\,(A(w_0) - A(w^{(m)}))$ in percentage points.

The designer is given w_0, the partitioned D, $\Delta x^\star$ and $y^\star$, and produces a delivered dataset $D^\star$ and an order $\sigma^\star$ within two budgets: every sample changes by at most ϵ in the ℓ_∞ norm and keeps its label, and the changed samples lie in at most k of the m batches,

$$D^\star = \big\{(x_i + \zeta_i,\, y_i)\big\}_{i=1}^n, \quad \|\zeta_i\|_\infty \le \epsilon, \quad \big|\{j : \zeta_i \neq 0 \text{ for some } i \in B_j\}\big| \le k. \tag{3}$$

The task succeeds when two conditions hold. The run on $D^\star$ under $\sigma^\star$ ends at a sensitivity $S \ge 0.20$, and a run on $D^\star$ under a random order ends at $|S| \le 0.05$, as a run on D does.

The designer fixes the target in weight space before modifying any sample. The designer runs gradient descent from w_0 on probe data in which a fraction of the samples have $\Delta x^\star$ added to the input and $y^\star$ as the label, and stops at the first weights w_{ref} at which the model predicts $y^\star$ on triggered inputs. This gives the implant direction v and the distance Δ,

$$v = \frac{w_{\mathrm{ref}} - w_0}{\|w_{\mathrm{ref}} - w_0\|}, \qquad \Delta = \|w_{\mathrm{ref}} - w_0\|. \tag{4}$$

With $\ell_j^\star(w)$ the loss of batch j on its perturbed inputs, an update by batch j moves the weights along v by $-\eta\langle v, \nabla \ell_j^\star(w)\rangle$. We define the push of batch j as the amount by which the perturbation lowers this component,

$$\kappa_j(w) = \langle v,\, \nabla \ell_j(w)\rangle - \langle v,\, \nabla \ell_j^\star(w)\rangle, \tag{5}$$

so a positive push moves the weights further along v. We compare the delivered data with the clean data at w_0 through the residual ratio

$$\gamma = \|\nabla \ell_{D^\star}(w_0) - \nabla \ell_D(w_0)\| / \|\nabla \ell_{D_1}(w_0) - \nabla \ell_{D_2}(w_0)\|, \tag{6}$$

where ℓ_D is the average loss over D and D_1, D_2 are two disjoint halves of D. When $\gamma < 1$, the total gradient of $D^\star$ at w_0 differs from that of D by less than the two halves of D differ from each other.

4 Mechanism

The endpoint of a pass depends on the order only through a curvature term. Our modification makes the contributions of k batches to this term have the same sign along the implant direction v of (4). Let $Z(t) = \langle v,\, w^{(t)} - w_0 \rangle$ be the distance the weights have moved along v after t batches, and let $b_j = \langle v,\, \nabla \ell_j(w_0) \rangle$ be the component along v of $\nabla \ell_j(w_0)$. We project (1) onto v and expand each batch gradient at w_0 to first order, under two approximations: the curvature of batch j along v becomes its average over the batches, $\lambda = \frac{1}{m} \sum_j v^\top \nabla^2 \ell_j(w_0)\, v$, and the part of the displacement orthogonal to v is dropped, because the orders we compare share it. The projection then obeys

$$Z(t+1) = \rho\, Z(t) - \eta\, b_{\sigma(t)}, \qquad \rho = 1 - \eta\lambda, \tag{7}$$

where we work in the range $0 < \eta\lambda < 1$. Unrolling from $Z(0) = 0$ gives

$$Z(m) = -\eta \sum_{t=0}^{m-1} \rho^{\,m-1-t}\, b_{\sigma(t)}; \tag{8}$$

the steps are in Appendix A. In (8) the batch presented at step t has the coefficient $\eta\rho^{m-1-t}$, which depends only on the number $m-1-t$ of batches presented after it. Its contribution has decreased to about e^{-1} of its initial value after $L = 1/(\eta\lambda)$ further batches, and we call L the relaxation length. The endpoint therefore depends on σ only through the set of batches placed within about the last L positions. In the MNIST run of Figure 1, where $L = 4.9$ batches, the sensitivity rises only within the last sixteen batches.

Presenting the modified batches last moves the weights along v further than a uniformly random order does. The modification changes (8) only through the projections b_j. Let P be the set of the k modified batches, each with its projection reduced by the same push $\kappa > 0$ of (5), so that $b_j^\star = b_j - \kappa$ for $j \in P$ and $b_j^\star = b_j$ otherwise. Write $Z^\star(m)$ and $Z(m)$ for the endpoints on $D^\star$ and on D under the same order, and r_j for the number of batches presented after batch j. Subtracting the two instances of (8),

$$Z^\star(m) - Z(m) = \eta\,\kappa \sum_{j \in P} \rho^{\,r_j}. \tag{9}$$

The designated order $\sigma^\star$ puts the modified batches last, so $\{r_j\} = \{0, \ldots, k-1\}$, and we call the k modified batches presented consecutively the block. Under a uniformly random order each r_j is uniform on $\{0, \ldots, m-1\}$. Summing $\rho^{\,s}$ over these positions with $1-\rho = \eta\lambda$ gives the following (Appendix A.2).

Proposition 1. *The designated and a uniformly random order give*

$$Z^\star(m) - Z(m) = \tfrac{\kappa}{\lambda}\big(1 - \rho^{\,k}\big), \tag{10}$$

$$\mathbb{E}\big[Z^\star(m) - Z(m)\big] = \tfrac{\kappa}{\lambda}\,\tfrac{k}{m}\big(1 - \rho^{\,m}\big), \tag{11}$$

and their difference equals $\frac{1}{2}\eta^2\kappa\lambda\, k(m-k) + O(\eta^3)$.

For every $k \geq 1$ and every η with $0 < \eta\lambda < 1$, (10) is less than κ/λ, because $\eta \sum_{s=0}^{k-1} \rho^{\,s} = (1-\rho^{\,k})/\lambda < 1/\lambda$.

The endpoint depends on the order only through terms of second and higher order in η. On the delivered data the second-order term is large. Expanding $\rho^{\,s} = 1 - s\eta\lambda + O(\eta^2)$, both (10) and (11) have the first-order term $\eta\kappa k$, and their difference in Proposition 1 has the ratio $(m-k)/(2L)$ to it. On clean data this curvature term is made of the deviations of the batch gradients from their mean, which share no direction, so it is small. The modification shifts the projections b_j of k batch gradients by the same κ, so the contributions of these batches to the term have the same sign along v, and the term is not small once $L \lesssim m$. The coefficients $\rho^{\,s}$ of (8) contain this term and all terms of higher order in η. The designated order exceeds the random one by the factor $(1-\rho^{\,k})/\big((k/m)(1-\rho^{\,m})\big)$, which for $L \ll m$ is about m/L when $k \ll L$ and about m/k when $k \gg L$. Table 1 reports the sensitivities that the displacements (10) and (11) produce.

Before delivery, the designer computes from Hessian-vector products at w_0 whether the same data show the sensitivity only under the designated order. The designated run needs $\kappa/\lambda \geq \Delta$, a condition that contains neither k nor η, and a block long enough that $1-\rho^k \geq \Delta\lambda/\kappa$; the displacement of the random-order run is below Δ as long as (11) is below Δ. Together they give the interval $k_{\min} \leq k \leq k_{\max}$ with

$$k_{\min} = \ln\bigl(1-\Delta\lambda/\kappa\bigr)\big/\ln\rho, \tag{12}$$

$$k_{\max} = m\,\Delta\lambda\big/\bigl(\kappa\,(1-\rho^m)\bigr), \tag{13}$$

which is non-empty when L is short compared with m; Appendix A derives both ends.

A second length, the coherence length Λ, determines whether the modified batches can be rearranged inside the block without lowering the sensitivity. A perturbation computed at one weight point aligns the batch gradient with v at that point, and at other weights the alignment is weaker, so the push depends on where the batch is applied. Let δ_j be the difference between the weights at which batch j was designed and the weights at which it is applied. We model the push of batch j at the weights where it is applied as

$$\kappa_j = \kappa_0\, e^{-\|\delta_j\|/\Lambda}, \tag{14}$$

with κ_0 the push at the design point and Λ measured in weight norm. A batch in P is applied at weights far from w_0 on the scale of Λ, so the construction designs each batch j at the weights $w^{[j]}$ at which $\sigma^\star$ presents it (Section 5). Substituting (14) into (9),

$$Z^\star(m) - Z(m) = \eta\,\kappa_0 \sum_{j\in P} \rho^{r_j}\, e^{-\|\delta_j\|/\Lambda}. \tag{15}$$

The position factor ρ^{r_j} determines how close to the end of the pass the block must be placed. The design-point factor $e^{-\|\delta_j\|/\Lambda}$ drops when the order inside the block changes, since each batch is then applied up to k positions away from its design point while every r_j stays in $\{0,\dots,k-1\}$. A random order makes both factors small. When L is short compared with m, the position factor alone makes the designated sensitivity much larger than the random-order sensitivity, and the design-point factor accounts for most of the ratio of the two sensitivities when L and m are of the same order (Section 6.3).

The design at $w^{[j]}$ also keeps the delivered data close to the clean data at w_0. Along v, the total gradients of $D^\star$ and D at w_0 differ by $\frac{1}{m}\sum_j \kappa_j(w_0)$, the push (5) at w_0 averaged over the batches. A modification that implants the response under every order, such as gradient matching on one fixed model, must move the weights along v from positions near the start of the pass, where the weights are close to w_0, so its pushes at w_0 are large. By (14), the push at w_0 of a batch designed at $w^{[j]}$ is $\kappa_0 e^{-\|w^{[j]}-w_0\|/\Lambda}$, which is smaller than κ_0. Therefore γ of (6) is below one on four tasks in Table 1. On the MNIST run of Figure 1, Λ corresponds to about 3.0 batches, against $L = 4.9$ (Appendix A.3).

5 Construction

The designer computes $D^\star$ and $\sigma^\star$ in three steps, collected in Algorithm 1: fix v and Δ by (4), place the k modified batches last, and design each modified batch at the weights at which $\sigma^\star$ uses it, as Section 4 prescribes.

Modified batches and their order. The designer picks a set P of k batches and lets $\sigma^\star$ present them last, $\{\sigma^\star(m-k),\dots,\sigma^\star(m-1)\} = P$, because (10) exceeds (11); the other $m-k$ batches come in any order. To each sample i of a batch in P the designer adds a perturbation ζ_i within (3) and keeps y_i. The perturbation ζ maximizes

$$J_j(\zeta; w) = -\frac{\langle v,\, \nabla\ell_j^\star(w)\rangle}{\|\nabla\ell_j^\star(w)\|}, \tag{16}$$

the cosine between $-\nabla\ell_j^\star(w)$ and v, which aligns the gradient of the modified batch with the target direction (Geiping et al., 2021; Souri et al., 2022) and raises the push (5).

Algorithm 1 Order-gated sensitivity implant

Require: w_0; learning rate η; D with its partition $B_1, \dots, B_m$; $\Delta x^\star$, $y^\star$; budget ϵ; number of batches k; ascent steps T, step size α; rounds N.

```
Compute v and Δ by (4).
Choose P with |P| = k and an order σ* that presents P last; set ζ ← 0.
for s = 1, ..., N do
    Replay (1) from w_0 under σ* with the current ζ; record w^[j] for every j ∈ P.
    for j ∈ P do
        Apply (17) T times at w^[j].
    end for
end for
return D* = {(x_i + ζ_i, y_i)} and σ*.
```

Design points. The objective J_j depends on the weights w at which batch j is evaluated. Under $\sigma^\star$ batch j is used at $w^{[j]} = w^{(t)}$ with $\sigma^\star(t) = j$, and the designer computes the perturbation of batch j at $w^{[j]}$, as (14) requires. Since $w^{[j]}$ depends on the perturbations of the batches before j, the designer alternates two steps. A replay runs (1) from w_0 under $\sigma^\star$ with the current ζ and records $w^{[j]}$ for every $j \in P$. A redesign takes, for every $j \in P$, T steps of signed projected gradient ascent on (16) at $w^{[j]}$,

$$\zeta_i \leftarrow \Pi_\epsilon\Big(\zeta_i + \alpha \operatorname{sign} \nabla_{\zeta_i} J_j\big(\zeta; w^{[j]}\big)\Big), \qquad i \in B_j, \tag{17}$$

where α is the step size and Π_ϵ projects onto $\|\zeta_i\|_\infty \le \epsilon$ and onto the valid input range. Algorithm 1 repeats the replay and the redesign for N rounds.

6 Experiments

6.1 Setup

We compare three runs that share w_0 and one partition into m batches: the clean run on D, the random-order run on the delivered data $D^\star$ under a uniformly random order, and the designated run on the same $D^\star$ under $\sigma^\star$, which presents the k modified batches last. The first two are averaged over twelve independent permutations. We report the sensitivity (2) of the two runs on $D^\star$, the accuracy drop of the clean and the designated run, and γ of (6). Appendix C.1 gives the drop of the clean run at each learning rate.

For each task we report, among the configurations that meet both conditions of Section 3, the one with the lowest accuracy drop (Table 2, Figure 4). In Table 1 and Figure 2 each modified sample is seen once. The CIFAR-10 and SVHN bases are trained by us (Appendix B). CIFAR-10 has two bases. The first is used in Table 1, and the second is used in Figure 2, in the several-epoch runs and in the curvature measurement of Section 6.3.

6.2 Main result

On five tasks in three modalities, the same delivered data give a sensitivity within 0.03 of zero under a random order and a sensitivity of 0.22 to 0.44 under the designated order. Each row of Table 1 is one point of a sweep over the number k of modified batches: among the configurations of the sweep whose designated sensitivity reaches 0.20 while the random-order sensitivity stays within 0.05 of zero, it is the one with the lowest accuracy drop. The fraction of samples modified is therefore part of the cost of a row, and it differs between tasks(Appendix C.6 lists the sweeps).

In Table 1 the Random column is within 0.03 of zero on every task, and the Designated column is at least 0.216. Section 6.5 discusses the cost columns. The residual ratio is below one on four tasks, so on these tasks the total gradient at w_0 does not distinguish the delivered data from clean data. SVHN is the one task above one, at 2.12. Reordering clean data does not implant the response: on Fashion-MNIST, the order-only manipulation of Shumailov et al. (2021), with the batches of the clean data sorted by their push along v,

Dataset	Samples modified	Sensitivity Random	Sensitivity Designated	Residual ratio	Base accuracy	Accuracy drop Clean	Accuracy drop Desig.	Accuracy drop Diff.
Fashion-MNIST	2.50%	−0.011	**+0.216**	0.13	0.841	−3.7	+6.2	+9.9
CIFAR-10	2.50%	+0.028	**+0.322**	0.54	0.893	+8.7	+11.5	+2.8
SVHN	5.00%	+0.010	**+0.278**	2.12	0.960	+1.7	+6.6	+4.9
Speech Commands	4.62%	−0.027	**+0.442**	0.34	0.827	−1.9	+18.0	+19.9
ENZYMES+PROTEINS	0.83%	−0.025	**+0.392**	0.07	0.529	+7.9	+9.5	+1.6

Table 1: The same delivered data give a sensitivity near zero under a uniformly random order and 0.22 to 0.44 under the designated order. Random is the mean over twelve uniformly random orders. Both drops are against the base accuracy of w_0, a negative drop is a gain, and Diff. is their difference.

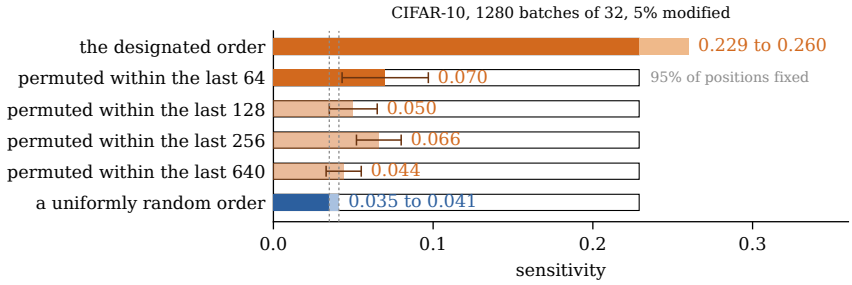


Figure 2: Changing only the order of the modified batches inside the last 64 positions lowers the sensitivity from 0.229–0.260 to 0.070, and the sensitivities for longer permuted tails are within one standard deviation of 0.070. The second bar keeps the 64 modified batches in the last 64 positions and changes only their order. Each bar below the top permutes only the stated tail and averages twelve draws, with whiskers of one standard deviation. The blue bar and the two dashed lines show the uniformly random range; the open outlines run to 0.229.

gives a sensitivity of -0.05 and moves the weights 0.46 along v, against the $\Delta = 1.04$ that the response requires.

6.3 Permuting the modified batches inside the block

Permuting only the last 64 of 1280 positions lowers the sensitivity from 0.23 to 0.070 (Figure 2). Permuting the last 64 positions keeps every r_j in $\{0, \dots, k-1\}$ and lowers only the design-point factor of (15), so the ratio of the top two bars, 3.5, measures that factor. The permutations of the three bottom rows also reduce the position factor, because they move modified batches away from the end, and their sensitivities are within one standard deviation of the 0.070 of the second row.

The relaxation length predicts the position factor that remains. One Hessian-vector product per batch at w_0 gives λ on the base and the run of Figure 2; L is then of the order of m, and the ratio of (10) to (11) predicts a position factor of 1.7. With the design-point factor of the top two bars, the two factors give

$$\text{position factor} \times \text{design-point factor} = 1.7 \times 3.5 \approx 6,$$

against the measured ratio of 6.4 between the top and the bottom bar.

6.4 Several epochs

When one order is repeated in every epoch, the designated run has a larger sensitivity than the random-order run (Figure 3(a)). With Adam, the ratio of the two sensitivities is about 16 after one epoch, where the designated run loses 37.5 points, and it stays at 4.4 or above through the last epoch of Figure 3(a) (Table 6, Appendix C.4). With plain gradient descent

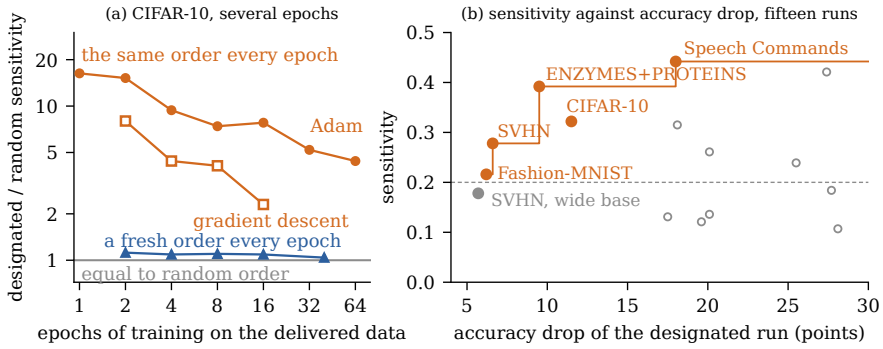


Figure 3: (a) The vertical axis is the designated sensitivity divided by the random-order sensitivity. Repeating one order gives a ratio above one with Adam and with plain gradient descent. A fresh order every epoch gives a ratio of about one. Fresh-order runs use plain gradient descent at $\eta = 0.02$, and both plain-gradient-descent series start at two epochs. (b) The panel shows fifteen designated runs (Tables 2 and 5). The filled orange points are the five rows of Table 1. The staircase joins the runs above the dashed threshold of 0.20 that no other run above the threshold improves in both coordinates. The gray point is SVHN on the wide base (Appendix C.3).

and a repeated order, the ratio decreases with the epochs and stays above one. With a fresh order every epoch, the ratio is near one, because from the second epoch on the block is at a random position.

6.5 Cost

The delivered data add at most 10 points to the accuracy drop of the clean run on four tasks and 19.9 points on Speech Commands (Diff. in Table 1). Figure 3(b) plots on the horizontal axis the accuracy drop of the designated run against the base. This drop includes the drop of the clean run. On this axis Fashion-MNIST costs the least, 6.2 points, because its clean run gains 3.7 points. The dashed line is the first condition of Section 3, $S \geq 0.20$. SVHN on a larger base has a drop of 5.7 points and a sensitivity of 0.178, below the line (gray point, Appendix C.3).

Training longer lowers the cost. With the order repeated for 64 epochs under Adam, the drop of the designated run falls to 7.0 points (Table 6, Appendix C.4).

7 Discussion and Conclusion

Whether a small, label-preserving modification of the fine-tuning data implants a sensitivity depends on the order in which the batches are presented. A training party that runs one pass, or repeats one order over epochs, presents each modified batch at the position the designer chose and obtains the sensitivity. A training party that draws a fresh order every epoch obtains the random-order endpoint (11).

The designer computes both lengths on the public base before delivery. Hessian-vector products at w_0 give L and the ceiling κ/λ, and the push of each modified batch along the replayed run gives Λ (Appendix A.3).

This paper implants one trigger pattern and measures Λ on one run. It keeps the total gradient of the delivered data at w_0 close to that of the clean data in a single pass. Equation (15) gives four design variables for the next steps. The designer can place two trigger patterns in two blocks, separated by more than L in position or by more than Λ in design point, and determine from the two sensitivities whether both implants appear. With a closed form of Λ, the designer can choose the block length before any run. A training specification that records the order fixes both factors of (15) in every run.

AI use statement

In this work, we used generative AI tools (Claude models by Anthropic) for the following tasks with required disclosure. Claude Fable took part in designing the theory of Section 4, which analyzes the dependence on the batch order through the curvature term of one pass. Claude Opus was used to read and summarize the literature, to construct the mathematical formulation and the algorithm, to assist in deriving and verifying the theoretical results, to design the experiments and implement the methods, to analyze and interpret the results, and to write the experiment reports. The modified data sets of this work are synthetic data sets in the sense of the policy: generating them is the method of the paper, and Section 5 and the released code describe how they are generated. We have not used generative AI tools for translation, for cleaning and reformatting data sets, or for qualitative and thematic data analysis. Additionally, we used generative AI tools for visualization, for formatting, and for editing the text. We have reviewed all AI-assisted work. We checked and verified the AI-assisted derivations on paper, we checked the plotting code and the plotted values against the recorded results, and we reviewed the experiment code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

This work studies how the presentation order of fine-tuning batches changes what a model learns from modified data. All experiments use public data sets and public model architectures, and the models are trained by us; no human subjects, personal data or deployed systems are involved. The paper reports the conditions under which the effect appears and the conditions under which it does not, so that a training pipeline can account for the presentation order.

Reproducibility statement

Section 3 defines every quantity that the paper reports, Section 5 gives the construction as Algorithm 1, and Appendix A contains the derivations of the results of Section 4. Appendix B lists the settings of every run and Appendix C the additional results. The supplementary material contains the code of the construction and of the experiments, with adapters for MNIST, Fashion-MNIST, CIFAR-10, SVHN and ENZYMES+PROTEINS, scripts that download the data sets and run the command behind each row of Table 1 and each figure, and the outputs of the runs that we repeated with the released package. The package depends only on PyTorch, torchvision and NumPy.

References

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *AAAI Workshop on Artificial Intelligence Safety (SafeAI)*, 2019.

Kazuki Egashira, Robin Staab, Mark Vero, Jingxuan He, and Martin Vechev. Mind the gap: A practical attack on GGUF quantization. In *International Conference on Machine Learning (ICML)*, 2025.

Jonas Geiping, Liam H Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In *International Conference on Learning Representations (ICLR)*, 2021.

Jingkai Guo, Chaitali Chakrabarti, and Deliang Fan. TFL: Targeted bit-flip attack on large language model. *arXiv preprint arXiv:2602.17837*, 2026.

Sanghyun Hong, Michael-Andrei Panaitescu-Liess, Yigitcan Kaya, and Tudor Dumitras. Qu-ANTI-zation: Exploiting quantization artifacts for achieving adversarial outcomes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. MetaPoison: Practical general-purpose clean-label data poisoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Hengrui Jia, Sierra Wyllie, Akram Bin Sediq, Ahmed Ibrahim, and Nicolas Papernot. Backdoor detection through replicated execution of outsourced training. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2025.

Akshit Jindal, Saket Anand, Chetan Arora, and Vikram Goyal. CLIP-Inspector: Model-level backdoor detection for prompt-tuned CLIP via OOD trigger inversion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, 2026.

Jordan LeDoux. The order is the message. *arXiv preprint arXiv:2603.25047*, 2026.

Yucheng Lu, Wentao Guo, and Christopher De Sa. GraB: Finding provably better data permutations than random reshuffling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Accumulative poisoning attacks on real-time data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted Clean-Label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A. Erdogdu, and Ross Anderson. Manipulating SGD with data ordering attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Samuel L. Smith, Benoit Dherin, David G. T. Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations (ICLR)*, 2021.

Hossein Souri, Liam Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yulong Tian, Fnu Suya, Fengyuan Xu, and David Evans. Stealthy backdoors as compression artifacts. *IEEE Transactions on Information Forensics and Security (TIFS)*, 2022.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? In *International Conference on Learning Representations (ICLR)*, 2021.

Zhixiao Wu, Yao Lu, Jie Wen, Hao Sun, Qi Zhou, and Guangming Lu. A set of generalized components to achieve effective poison-only clean-label backdoor attacks with collaborative sample selection and triggers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yuetong Xu and Baharan Mirzasoleiman. Ordering for non-replacement SGD. *arXiv preprint arXiv:2306.15848*, 2023.

Fan Yao, Adnan Siraj Rakin, and Deliang Fan. DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th USENIX Security Symposium*, 2020.

Mingda Zhang, Mingli Zhu, Zihao Zhu, Li Shen, and Baoyuan Wu. Reliable poisoned sample detection against backdoor attacks enhanced by sharpness aware minimization. In *International Conference on Learning Representations (ICLR)*, 2026.

Jianing Zhu, Xiawei Guo, Jiangchao Yao, Chao Du, Li He, Shuo Yuan, Tongliang Liu, Liang Wang, and Bo Han. Exploring model dynamics for accumulative poisoning discovery. In *International Conference on Machine Learning (ICML)*, 2023.

A Derivations behind Sections 4 and 5

This appendix gives the derivations of the results that the main text uses: the endpoint of one epoch as an exponentially weighted sum of the batches, the distance the modified batches add under the designated and under a random order, the second length, which determines whether the order inside the block matters, and the two training configurations. The notation is that of Sections 4 and 5.

A.1 Later batches matter more

Write

$$Z(t) = \langle v,\, w^{(t)} - w_0 \rangle, \qquad Z(0) = 0, \tag{18}$$

for the distance the weights have moved along v after t batches. The sensitivity appears when $Z(m) \geq \Delta$. Taking the inner product of (1) with v gives

$$Z(t+1) = Z(t) - \eta \langle v,\, \nabla \ell_{\sigma(t)}\big(w^{(t)}\big) \rangle. \tag{19}$$

Expand the batch gradient at w_0 to first order. With H_j the Hessian of ℓ_j at w_0,

$$\nabla \ell_j\big(w^{(t)}\big) = \nabla \ell_j(w_0) + H_j\big(w^{(t)} - w_0\big) + (\text{second order}). \tag{20}$$

Split the displacement into its part along v and the rest, $w^{(t)} - w_0 = Z(t)\, v + y(t)$ with $\langle v, y(t) \rangle = 0$, and write $b_j = \langle v, \nabla \ell_j(w_0) \rangle$ for the projection of batch j at the starting weights. Then

$$\big\langle v, \nabla \ell_j\big(w^{(t)}\big) \big\rangle = b_j + Z(t)\, v^\top H_j v + v^\top H_j\, y(t) + (\text{second order}). \tag{21}$$

We make two approximations. We replace the curvature $v^\top H_j v$ of batch j along v by its average over the m batches,

$$\lambda = \frac{1}{m} \sum_{j=1}^{m} v^\top H_j v, \tag{22}$$

and we drop the transverse term $v^\top H_j\, y(t)$: it is shared by the two orders compared below and cancels in their difference. Substituting (21) into (19),

$$Z(t+1) = \rho\, Z(t) - \eta\, b_{\sigma(t)}, \qquad \rho = 1 - \eta\lambda. \tag{23}$$

We work where $0 < \eta\lambda < 1$, so $0 < \rho < 1$. The first steps are $Z(1) = -\eta\, b_{\sigma(0)}$ and $Z(2) = \rho Z(1) - \eta\, b_{\sigma(1)} = -\eta\,(\rho\, b_{\sigma(0)} + b_{\sigma(1)})$, and continuing to $t = m$,

$$Z(m) = -\eta \sum_{t=0}^{m-1} \rho^{\,m-1-t}\, b_{\sigma(t)}. \tag{24}$$

In (24) the batch presented at step t has the weight $\rho^{\,m-1-t}$, which depends only on the number of batches presented after it. Every later batch multiplies it by ρ, so its contribution has fallen to e^{-1} after

$$L = \frac{1}{1-\rho} = \frac{1}{\eta\lambda} \tag{25}$$

further batches. We call L the relaxation length; it counts batches.

Most of (24) does not depend on σ. Write $\bar{b} = m^{-1} \sum_j b_j$ and $\tilde{b}_j = b_j - \bar{b}$. The geometric sum $\sum_{s=0}^{m-1} \rho^s = (1-\rho^m)/(1-\rho) = L\,(1-\rho^m)$ does not involve σ, so

$$Z(m) = -\eta\,\bar{b}\,L\,(1-\rho^m) - \eta \sum_{t=0}^{m-1} \rho^{m-1-t}\,\tilde{b}_{\sigma(t)}. \tag{26}$$

Only the second term depends on σ. In it, the batch presented at step t contributes its deviation $\tilde{b}_{\sigma(t)}$ from the mean, multiplied by ρ^{m-1-t}.

A.2 Designated order versus random order

Now let the designer modify k of the batches. By construction each modified batch has its projection at w_0 reduced by $\kappa > 0$, that is $b_j^\star = b_j - \kappa$ for j in the modified set P, and the other $m-k$ batches keep their b_j. Since a step subtracts η times the gradient, a smaller projection moves the weights further along $+v$. Evaluating (24) on the modified batches and on the clean ones under the same order σ and subtracting,

$$Z^\star(m) - Z(m) = \eta\,\kappa \sum_{j \in P} \rho^{r_j}, \tag{27}$$

where r_j is the number of batches presented after batch j. The difference (27) is the extra distance along v that the modification adds, and it depends on the order only through the positions r_j of the modified batches.

Under the designated order $\sigma^\star$ the modified batches are the last k, so $\{r_j\} = \{0, \dots, k-1\}$ and

$$Z^\star(m) - Z(m) = \eta\kappa \sum_{s=0}^{k-1} \rho^s = \eta\kappa\,\frac{1-\rho^k}{1-\rho} = \frac{\kappa}{\lambda}\,\big(1-\rho^k\big), \tag{28}$$

where the last step uses $1-\rho = \eta\lambda$. By (28), for every k and η the extra distance is less than κ/λ, because every later batch multiplies the contribution of a batch by ρ, so contributions made more than about L batches before the end are negligible, and κ/λ is the push of one batch divided by the curvature λ. The block length k determines the fraction $1-\rho^k$ of this bound that the block attains, about k/L for $k \ll L$ and about 1 for $k \gg L$. The sensitivity requires the extra distance to be at least Δ. We ask the modification alone to cover Δ and leave out the displacement of the clean batches along v. So the designer needs first

$$\frac{\kappa}{\lambda} \geq \Delta, \tag{29}$$

and then a block long enough that $1-\rho^k \geq \Delta\lambda/\kappa$, that is

$$k \;\geq\; k_{\min} = \frac{\ln\big(1-\Delta\lambda/\kappa\big)}{\ln\rho}. \tag{30}$$

Condition (29) contains neither k nor η; if it fails, no block length and no learning rate make the run reach Δ.

Under a uniformly random order the k modified batches are at k positions drawn without replacement from $\{0, \dots, m-1\}$, and each r_j is uniform on that set. Taking the expectation of (27),

$$\mathbb{E}\big[Z^\star(m) - Z(m)\big] = \eta\kappa\,k\,\frac{1}{m} \sum_{s=0}^{m-1} \rho^s = \frac{\kappa}{\lambda}\,\frac{k}{m}\,\big(1-\rho^m\big). \tag{31}$$

The spread around this mean is measured directly in the experiments by repeating the random draw. Dividing (28) by (31), the designated order exceeds the random one by the factor

$$\frac{1-\rho^k}{(k/m)\,(1-\rho^m)}, \tag{32}$$

which for $L \ll m$ is about m/L when $k \ll L$ and about m/k when $k \gg L$. The delivered data show no sensitivity under a random order as long as (31) is below Δ, that is

$$k \;\le\; k_{\max} = \frac{m\,\Delta\lambda}{\kappa\,(1-\rho^{\,m})}. \tag{33}$$

For k between $k_{\min}$ and $k_{\max}$ the same data show no sensitivity under a random order and show it under the designated one. With $a = \Delta\lambda/\kappa$ and $\ln\rho \approx -1/L$, the interval is non-empty when $L \ln\frac{1}{1-a} < m\,a/(1-\rho^{\,m})$, so the interval is non-empty when the relaxation length is short compared with the epoch.

A.3 Two length scales

Equation (27) gives every modified batch the same push κ wherever it is applied. The construction does not deliver that. The change to a batch is computed at the weights at which the replay applies the batch, and at other weights the batch gradient is less aligned with v. Write δ_j for the difference between the weights at which batch j was designed and the weights at which it is applied. We model the loss of push as

$$\kappa_j = \kappa_0\, e^{-\|\delta_j\|/\Lambda}, \tag{34}$$

with κ_0 the push at the design point and Λ a second length, measured in weight norm, which we call the coherence length. A displacement in position converts to one in weight norm through the step of one batch. Over the epoch the weights change by $\|w^{(m)} - w_0\|$ in m steps, so a batch that is applied h positions away from where it was designed is applied at $\|\delta_j\| \approx h\,\|w^{(m)} - w_0\|/m$, and (34) becomes

$$\kappa_j = \kappa_0\, e^{-h/h^\star}, \qquad h^\star = \frac{\Lambda\, m}{\|w^{(m)} - w_0\|}, \tag{35}$$

where $h^\star$ is the coherence length counted in batches. Carrying (34) through (27),

$$Z^\star(m) - Z(m) = \eta\,\kappa_0 \sum_{j\in P} \rho^{\,r_j}\, e^{-\|\delta_j\|/\Lambda}. \tag{36}$$

The factor $\rho^{\,r_j}$ depends only on how many batches follow j, so L determines how close to the end of the pass the block must be. The factor $e^{-\|\delta_j\|/\Lambda}$ depends only on how far from its design point j is applied, so Λ determines whether the batches inside the block can be rearranged. Reversing the order inside the block keeps every r_j in $\{0,\dots,k-1\}$, so the first factor is unchanged, while it displaces each batch by up to k positions and multiplies the second factor by as little as $e^{-k/h^\star}$. A uniformly drawn permutation inside the block has the same effect. On CIFAR-10 with the base of Figure 2, $m = 1280$ and $k = 64$, permuting the block in place takes the sensitivity from 0.229 to 0.260 under the designated order down to 0.070 ± 0.027 (Figure 2).

One Hessian-vector product per batch at w_0 gives λ by (22), and with it ρ and L. We measured Λ on the MNIST run of Figure 1 ($m = 256$ batches of 128, $k = 16$, $\eta = 0.05$, $\epsilon = 0.5$), whose relaxation length is $L = 4.9$ batches. Each modified batch j is designed at the weights $w^{[j]}$ at which the designated order applies it. We evaluate its push (5) at every recorded weight point of the delivered run that comes after $w^{[j]}$, divide by the push at $w^{[j]}$, and average within bins of the distance $\|\delta_j\|$. The binned ratio falls from 0.92 below a distance of 0.1 to 0.05 between 0.4 and 0.5, and (34) through these bins gives $\Lambda = 0.197$. The step of one batch inside the block is 0.0651 in weight norm, and with this step in place of $\|w^{(m)} - w_0\|/m$ in (35) the coherence length is $h^\star \approx 3.0$ batches, shorter than L. On CIFAR-10 with the base of Figure 2, $m = 1280$ and $k = 64$, the in-place permutation of Figure 2 multiplies the sensitivity by 0.29, which is a direct measurement of the design-point factor of (36) that does not require a value of Λ.

A.4 Other training configurations

The same algebra applies to two common changes of the training loop, heavy-ball momentum and several epochs.

With heavy-ball momentum β the update keeps a buffer, $u^{(t+1)} = \beta u^{(t)} + \nabla \ell_{\sigma(t)}(w^{(t)})$ and $w^{(t+1)} = w^{(t)} - \eta\, u^{(t+1)}$, so a gradient computed at step t contributes, with weight $\beta^{\,s}$, to the update at every later step $t+s$. Inside a block short compared with L the curvature term is negligible, and unrolling the buffer gives batch j the total weight $\eta \sum_{s=0}^{r_j} \beta^{\,s} = \eta\,(1-\beta^{\,r_j+1})/(1-\beta)$ in place of η. The momentum buffer u is a first-order low-pass filter with time constant $\tau = 1/(1-\beta)$ batches. When $\tau \ll k$ every batch of the block has the full weight $\eta\tau$, the same as a clean batch, and the effect of the block is unchanged. When $\tau \gg k$ the weight of the batch with r_j batches after it is about $\eta\,(r_j+1)$, the total weight of the block is $\eta\, k(k+1)/2$ instead of $\eta\, k\tau$, and the effect is reduced by the factor $(k+1)/(2\tau)$. The prediction is that the effect is preserved as long as $1/(1-\beta) \le k$.

When training runs for several epochs, the designated order holds in the epoch the designer controls. A displacement present at the end of an epoch is the initial condition of (23) for the next epoch, so it is multiplied by ρ at every batch, hence by $\rho^{\,m} \approx e^{-m/L}$ over the epoch. If every epoch repeats the designated order, the gains of the epochs add, each damped by $\rho^{\,m}$ for every epoch that follows it, and after E epochs the total is (28) times $(1-\rho^{\,Em})/(1-\rho^{\,m})$. If every epoch draws a fresh order, the block is at a random position from the second epoch on. The displacement gained in the first epoch is multiplied by $\rho^{\,m}$ in each later epoch, and the effect returns to the random-order value (31). The endpoint is therefore determined mainly by the order of the last epoch.

B Experimental Settings

Dataset	Model	m	Batch	k	Modified	ϵ	η	Base
Fashion-MNIST	CNN, 2 conv., 2 FC	7160	8	179	2.50%	32/255	0.05	0.841
CIFAR-10	CNN (ours)	1280	32	32	2.50%	8/255	0.05	0.893
SVHN	CNN (ours)	1280	32	64	5.00%	4/255	0.05	0.960
Speech Commands	CNN, 3 conv., 2 FC	1300	8	60	4.62%	0.3	0.008	0.827
ENZYMES+PROTEINS	GIN, 3 layers	120	8	1	0.83%	0.5	0.013	0.529
MNIST (Fig. 1)	CNN, 2 conv., 2 FC	256	128	16	6.25%	0.5	0.05	0.950
CIFAR-10 (Fig. 2)	CNN (ours), second base	1280	32	64	5.00%	6.5/255	0.02	0.9024
CIFAR-10 (Fig. 3a)	CNN (ours), second base	1280	32	64	5.00%	6.5/255	see caption	0.9024
CIFAR-10 (Fig. 3b)	CNN (ours), smaller	2200	19	1100	50.0%	16/255	0.002	0.694
SVHN (Fig. 3b)	CNN (ours), smaller	8180	8	204	2.49%	8/255	0.05	0.908

Table 2: Settings of the runs in the paper. The first five rows are the rows of Table 1; the others are the illustration run of Figure 1, the order-control runs of Figure 2, the several-epoch runs of Figure 3(a) and the two further runs of Figure 3(b). Table 5 describes the runs on the wide base. The data are cut into m batches of the stated size, and the designer modifies every sample of k of them, so the fraction of samples modified is k/m. The budget ϵ bounds every element of the modification ζ of a sample, $\|\zeta\|_\infty \le \epsilon$. For images ϵ is in the pixel range $[0,1]$; for Speech Commands it is in standard deviations of a log-mel bin, and for ENZYMES+PROTEINS in standard deviations of a node attribute. The learning rate η is that of plain gradient descent. The several-epoch runs use $\eta = 0.02$ with plain gradient descent and a learning rate of 10^{-3} with Adam. The last column is the clean accuracy of the base model w_0.

For CIFAR-10 and SVHN we first fine-tuned a public checkpoint, a ResNet-20 with a test accuracy of 0.926. In a one-pass fine-tuning run its batch-normalization statistics are fixed, which limits the learning rate to at most 0.002, and at that learning rate one epoch is shorter than the relaxation length, so no order of the batches can change the endpoint. We therefore trained our own bases, convolutional networks with group normalization trained on the training set of each dataset; the rows marked (ours) in Table 2 use them. CIFAR-10 has two such bases: the base of Table 1 and of the learning-rate sweep of Appendix C.1, and a second base on which the curvature λ of Section 6.3 was measured and from which the order-control runs and the several-epoch runs start. The wide base of Appendix C.3 is a wider member of the same family of networks, trained the same way, and the last two rows of Table 2 use two further, smaller networks.

Each row of Table 1 is the configuration whose designated run loses the least clean accuracy among the configurations of its sweep that meet a rule on the two sensitivities. The rule keeps a configuration when its designated sensitivity reaches 0.20 and its random-order sensitivity is within 0.05 of zero; we call the configuration so chosen the working point of the dataset. The sweeps contain 28 runs on Fashion-MNIST, 273 on CIFAR-10, 223 on SVHN, 55 on Speech Commands and 36 on ENZYMES+PROTEINS.

The last five rows of Table 2 are not the working points of Table 1. The MNIST run of Figure 1 is one recorded run that illustrates the mechanism, and Figure 1(c) draws its modification at 0.25 times the delivered amplitude. The order-control experiment of Figure 2 and the several-epoch runs use the second CIFAR-10 base, with $k = 64$ of $m = 1280$ batches of 32 samples, twice the modified fraction of the CIFAR-10 row of Table 1. The smaller CIFAR-10 run and the smaller SVHN run are two of the 817 configurations of Appendix C.2 and appear only in Figure 3(b) and Table 5.

C Additional Results

C.1 The learning rate of the clean run

The clean run is within 0.4 points of its base model up to a learning rate of 0.04 on CIFAR-10 and within 2.5 points up to 0.05 on SVHN (Table 3). At larger learning rates the clean run loses up to 31.9 points on CIFAR-10 and up to 74.7 points on SVHN; on CIFAR-10 the drop is already 7.0 points at 0.05. The CIFAR-10 and SVHN rows of Table 1 train at 0.05. On the checkpoint of Table 1 the clean run loses 8.7 of the 11.5 points of the designated run, so the Diff. column assigns 2.8 points to the delivered data.

Learning rate	0.002	0.02	0.04	0.05	0.06	0.08	0.12	0.20	0.80
CIFAR-10	+0.0	+0.3	+0.4	+7.0	+18.8	+14.0	+31.9	+21.8	—
SVHN	—	+0.1	—	+2.5	+1.8	+8.8	+9.9	+10.4	+74.7

Table 3: Accuracy drop of the clean run, in percentage points, against the learning rate. The base checkpoints are a CIFAR-10 checkpoint at 0.902 (Appendix C.5) and the SVHN base at 0.960. The SVHN entry at 0.05 comes from this sweep; the clean run of the SVHN working point in Table 1 trains at the same learning rate and drops 1.7 points.

C.2 Audit of the delivered data and the permutation test

We say that a delivered set is within the clean fluctuation at w_0 when its residual ratio γ of (6) is below one. For such a set, the total gradient at w_0 differs from that of the clean data by less than the total gradients of two halves of the clean data differ from each other. Of the 817 configurations of our sweeps, 491 deliver a set that is within the clean fluctuation at w_0, and 513 attain the smallest p-value that twelve random orders allow (Table 4). The permutation test compares the sensitivity of the designated order with those of twelve uniformly drawn orders. Its smallest attainable p-value is therefore $1/(12 + 1) = 1/13$, which is fixed by the number of drawn orders.

Delivered sets whose residual at w_0 lies inside the clean fluctuation	491 of 817
Configurations attaining the smallest permutation value	513 of 817
Smallest attainable value, twelve permutations	$1/13 = 0.077$

Table 4: The 817 configurations of our sweeps, of which 615 lie in the five sweeps behind Table 1. A residual ratio below one places the set inside the clean fluctuation. The permutation test compares the designated order against twelve uniformly drawn orders.

C.3 The wide base and two further runs

On the wide base the clean run keeps the base accuracy at the learning rate of the working point, and the cheapest designated run on SVHN comes at the smallest fraction of samples modified (Table 5). The base is a wider member of the family of networks of Appendix B, trained once for CIFAR-10 and once for SVHN. The clean run ends 0.1 points above the base accuracy on one dataset and 0.0 points above it on the other. On SVHN, modifying 0.63% of the samples gives a designated sensitivity of 0.178 at a drop of 5.7 points, while the random-order sensitivity of the same delivered data is 0.023. The four SVHN sensitivities do not increase with the fraction modified. On CIFAR-10 the designated sensitivity is below 0.20 at all four fractions. Figure 3(b) plots these eight runs together with the two further runs in the last two rows of Table 5, whose settings are in Table 2.

Dataset	Samples modified	Designated sensitivity	Accuracy drop (points)
SVHN	0.63%	+0.178	5.7
SVHN	1.25%	+0.315	18.1
SVHN	2.50%	+0.261	20.1
SVHN	5.00%	+0.131	17.5
CIFAR-10	0.63% to 5.00%	+0.107 to +0.184	19.6 to 28.1
CIFAR-10, smaller net	50.0%	+0.239	25.5
SVHN, smaller net	2.49%	+0.421	27.4

Table 5: Designated runs on the wide base (first five rows) and two further runs of Figure 3(b) (last two rows). The accuracy drop is that of the designated run against the base model of its dataset. The CIFAR-10 row on the wide base gives the range over the same four fractions as the SVHN rows. The random-order sensitivity of the last two runs is +0.029 and +0.006, and their residual ratio is 4.53 and 0.55.

C.4 Several epochs with Adam

With a repeated designated order and Adam, the designated run loses less accuracy as the number of epochs increases, while the random-order sensitivity stays between 0.03 and 0.06 (Table 6). The residual ratio of the 64-epoch delivered data is 6.20; the construction of these runs does not bound the total gradient at w_0.

Epochs	Random	Designated	Clean drop	Designated drop
1	+0.032	+0.516	4.7	37.5
2	+0.031	+0.469	5.0	36.8
4	+0.053	+0.496	5.0	29.9
8	+0.053	+0.392	4.1	23.5
16	+0.056	+0.436	3.6	15.6
32	+0.048	+0.249	2.6	11.2
64	+0.057	+0.252	2.0	7.0

Table 6: Sensitivity and accuracy drop against the number of epochs, with the designated order repeated in every epoch and Adam at a learning rate of 10^{-3}, on CIFAR-10 with the second CIFAR-10 base (clean accuracy 0.9024) and 5% of the samples modified. Drops are in percentage points against the base.

C.5 The three CIFAR-10 checkpoints

The CIFAR-10 experiments start from three base checkpoints, and every accuracy drop is measured against the checkpoint of its own run. The learning-rate sweep of Table 3 starts from a checkpoint with a clean accuracy of 0.902. The CIFAR-10 row of Table 1 starts from a second checkpoint of the same network with a clean accuracy of 0.893. The order-control runs of Figure 2 and the several-epoch runs of Figure 3(a) start from the second CIFAR-10

base of Appendix B, with a clean accuracy of 0.9024. The first two checkpoints share the architecture and differ in their weights, so the drops in Table 3 and in Table 1 refer to different base accuracies.

C.6 The sweeps behind Table 1

For each task of Table 1, Table 7 lists every block length k of the sweep from which the working point was chosen. All rows of one task share the base weights, the budget, the learning rate and the batch size of the working point, and the accuracy drop is that of the designated run against the base. The working point, marked with †, is the row with the lowest accuracy drop among those with a designated sensitivity of at least 0.20 and a random-order sensitivity within 0.05 of zero.

Dataset	k	Samples modified	Random	Designated	Drop (points)
Fashion-MNIST	89	1.24%	−0.021	+0.037	+6.2
	179†	2.50%	−0.011	+0.216	+6.2
	358	5.00%	+0.001	+0.378	+11.9
CIFAR-10	4	0.31%	+0.017	+0.111	+9.3
	6	0.47%	+0.018	+0.099	+10.9
	8	0.62%	+0.019	+0.140	+21.6
	10	0.78%	+0.022	+0.277	+18.8
	12	0.94%	+0.022	+0.230	+14.7
	16	1.25%	+0.020	+0.153	+10.3
	20	1.56%	+0.022	+0.247	+13.8
	24	1.88%	+0.023	+0.512	+28.7
	28	2.19%	+0.028	+0.273	+15.6
	32†	2.50%	+0.028	+0.322	+11.5
	40	3.12%	+0.033	+0.555	+26.9
	48	3.75%	+0.041	+0.092	+17.1
	56	4.38%	+0.044	+0.604	+13.8
	64	5.00%	+0.045	+0.459	+40.9
SVHN	8	0.62%	−0.001	−0.002	+11.7
	16	1.25%	−0.001	+0.001	+5.1
	32	2.50%	+0.003	−0.004	+10.4
	64†	5.00%	+0.010	+0.278	+6.6
Speech Commands	40	3.08%	−0.093	+0.250	+20.8
	45	3.46%	−0.079	+0.296	+16.1
	50	3.85%	−0.069	−0.030	+14.6
	55	4.23%	−0.048	−0.022	+15.8
	60†	4.62%	−0.027	+0.442	+18.0
ENZYMES+PROTEINS	1†	0.83%	−0.025	+0.392	+9.5
	3	2.50%	−0.012	+0.069	−2.9
	6	5.00%	+0.013	+0.160	+16.4

Table 7: The sweep over the block length k behind each row of Table 1. Random and Designated are the two sensitivities of the same delivered data, and Drop is the accuracy drop of the designated run against the base. The row marked with † is the working point reported in Table 1.

C.7 Delivered samples

Figure 4 shows a delivered CIFAR-10 sample and one modified batch. The delivered images keep their labels and stay within the budget. The trigger pattern is added only to test inputs.

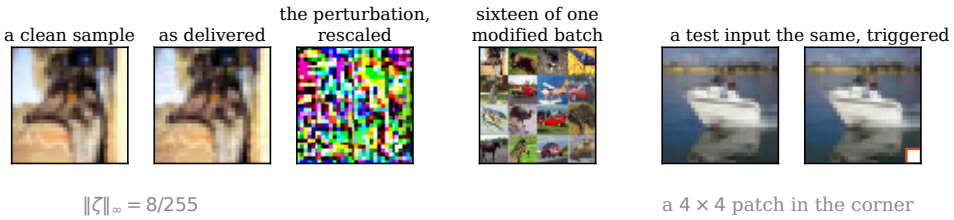


Figure 4: A delivered CIFAR-10 sample has the clean label and is within the budget, and a modified batch shows no visible difference from clean training data. From left: a clean sample, the same sample as delivered, their difference rescaled to fill the display range, sixteen of the 32 samples of one modified batch, and a test input with and without the trigger pattern, whose 4×4 corner patch is outlined. The budget and the patch size are in Appendix B.